

PMATH Reference Manual

Table of Contents

Preface	5
Introduction	6
Background	6
Availability	6
Design Principles	7
Scope and Naming Conventions	7
Random Number Generators	7
Constants	8
PMATH Routines by Category with MATHLIB Names	9
MATHLIB Routines Included in PMATH	9
Elementary Functions	9
Random Number Generators	9
Maxima and Minima	9
Table Look-Up Routines	9
Elementary Statistical Routines	9
Linear Algebra Routines	10
Root Finders	10
Interpolation and Approximation Routines	10
Differential Equation Solvers	10
Miscellaneous Routines	10
Error Procedures	11
MATHLIB Routines Omitted from PMATH	12
Elementary Functions	12
Random Number Generators	12
Maxima and Minima	12
Differential Equation Solvers	13
Error Procedures	13
Conversion of MATHLIB to PMATH Names	14
PMATH Routines by Category with PMATH Names	16
PMATH Routines Grouped by Function	16
Elementary Functions	16
Random Number Generators	16
Maxima and Minima	16
Table Look-Up Routines	17
Elementary Statistical Routines	17
Linear Algebra Routines	17
Root Finders	18
Interpolation and Approximation Routines	18
Differential Equation Solvers	18
Miscellaneous Routines	18
Error Procedure	18
Routines New to PMATH Explained	19
Vectorized RANF	19

Portable Seed-Passing Module	19
PMATH Routine Descriptions	21
AAAAAA	22
CV16TO64	23
CV64TO16	24
DCONST	25
DCORRV	27
DCOVAR	28
DFITPO	30
DLSODE	35
DMAXAF	55
DMEANF	56
DMEANV	57
DMEDF	58
DMINAF	59
DMINMX	60
DRANF	61
DRANFV	62
DRANKS	63
DREFIT	64
DRLGF	66
DSRCOM	67
DSTDEV	68
DUMACH	69
DZERO	70
IMAXAF	72
IMINAF	73
IMINMX	74
IUMACH	75
LDFD	76
LDFS	77
LUFd	78
LUFS	79
LUGD	80
LUGS	81
RANF8	82
RLFCNT	83
RLGF8	84
RLMSET	85
RLSGET	87
RLSSET	88
RNFCNT	90
RNMSET	91
RNSGET	93
RNSSET	94
RUMACH	95

SCONST	96
SCORRV	98
SCOVAR	99
SFITPO	101
SLSODE	106
SMAXAF	126
SMEANF	127
SMEANV	128
SMEDF	129
SMINAF	130
SMINMX	131
SRANF	132
SRANFV	133
SRANKS	134
SREFIT	135
SRLGF	137
SSRCOM	138
SSTDEV	139
SZERO	140
XERROR	142
XERRWD	143
XERRWV	144
XSETF	145
XSETUN	146
Disclaimer	147
Keyword Index	148
Alphabetical List of Keywords	150
Date and Revisions	152

Preface

Function: The PMATH mathematics subroutine library (libpmath.a) is a portable version of the CAL-coded part of the former MATHLIB (libmath.a) library, with a few extra routines added. PMATH supplements but does not duplicate the SLATEC library.

Availability: A chart showing the comparative availability of PMATH and SLATEC appears in the introductory section (page 6) below. MATHLIB itself is no longer available.

Consultant: For help contact the LC customer service and support hotline at 925-422-4531 (open e-mail: lc-hotline@llnl.gov, secure e-mail: lc-hotline@pop.llnl.gov).

This manual was adapted from the PMATH specifications and other related explanatory files written by Frederick N. Fritsch.

Printing: The print file for this document can be found at:

on the OCF: <http://www.llnl.gov/LCdocs/pmath/pmath.pdf>

on the SCF: https://lc.llnl.gov/LCdocs/pmath/pmath_scf.pdf

Introduction

Background

Inspired by a user survey conducted in November, 1992, LC's former Mathematical Software Support group developed portable versions of the CAL-coded part of MATHLIB (/usr/local/lib/libmath.a on the former CRAYs). This machine-independent mathematics subroutine library is called PMATH (/usr/local/lib/libpmath.a). This document is the PMATH reference manual.

Later sections of this manual explain the design principles (including the name-choice principles) for the PMATH library and how they were implemented. Because of the close connection between MATHLIB and PMATH, we introduce the PMATH routines using their MATHLIB counterparts, and note which MATHLIB routines were omitted from PMATH. A conversion chart between the MATHLIB and PMATH names is included. The PMATH routines are also listed by functional group under their own names, with features of the new routines explained. The largest part of this manual by far is an alphabetical dictionary of PMATH routines and the descriptive prologs for each, including the calling sequence.

Availability

MATHLIB was available on LC's CRAY J90 computers until they retired in March, 2000. Because the PMATH, the former MATHLIB, and SLATEC libraries are conceptually related, this chart shows where all three are available (in /usr/local/lib):

	Compaqs	IBMs	Linux
	-----	----	-----
MATHLIB	no	no	no
PMATH	yes	no(*)	yes
SLATEC	yes	no(*)	no(*)
	-----	----	-----
(*)but is downloadable from LINMath			

Design Principles

Decisions reached at a series of meetings held in July through September, 1993, became the guiding principles behind the PMATH specifications.

Scope and Naming Conventions

Principle 1.1. All user-callable routines in PMATH shall have different names than their counterparts in the former Cray libmath.a. Exception: PMATH routines may call standard subsidiary modules by their standard names. This exception applies to LINPACK routines, such as SGEFA/SGESL, as well as the (former) SLATEC error handlers XERROR/XERRWV and the machine precision function RUMACH.

Principle 1.2. Every module processing real data shall have an S-version (single precision), a D-version (double precision), and a version (referred to as the "REAL*8 version") that produces a 64-bit result regardless of the basic precision of the system. (This version is identical to the S-version on the former Crays or to the D-version on the Sun, for example.) We view the S- and D-versions as being the "true" PMATH. Implemented via conditional compilation or some other mechanism, the "REAL*8 version" merely provides a convenient platform-independent naming convention. It generally does not have separate documentation.

Principle 1.3. PMATH will support only the default INTEGER type of the host platform.

Principle 1.4. There shall be no optional arguments. All PMATH routines must be called with the full argument list. This is the main reason for Principle 1.1 (to force users to examine all former MATHLIB calls).

Principle 1.5. PMATH shall not duplicate capabilities already provided by the portable SLATEC library. Thus the LINPACK routines required by some PMATH routines are not considered part of PMATH.

Random Number Generators

Principle 2.1. DRANF on 32-bit platforms (or SRANF on 64-bit platforms) shall exactly reproduce the former Cray RANF results. There shall be a single sequence for both SRANF and DRANF. There shall be only one set of set/get routines for this sequence. SRANF shall call DRANF and simply truncate the result to single precision, modifying it if necessary to avoid returning exactly zero or one. In the present implementation, both are separate "wrappers" for the same underlying generator RANF8. (SRANF does not currently include the boundary checks.)

Principle 2.2. PMATH shall contain a vectorized version of RANF. The user interface is described in a later section (page 19).

Principle 2.3. A portable version of RLGF and its support routines shall be included in PMATH. These should be independent of the RANF sequence, as on the former Crays, and be designed according to the same principles as in 2.1.

Principle 2.4. RNCOUNT and RLGCNT will be included only if they can be implemented easily and at negligible runtime cost. (They are both included in the present implementation.)

Principle 2.5. PMATH shall contain a portable module to facilitate the passing of random number generator seeds between different machines. The user interface is described in a later section (page 19).

Constants

Principle 3.1. A portable version of `CONSTANT` shall be included in `PMATH`. It should return the correctly rounded value (to the precision implicit in its name) for each constant. On consultation with potential users, it is sufficient that `CONSTANT` return one of the two machine numbers closest to the correct value.

PMATH Routines by Category with MATHLIB Names

MATHLIB Routines Included in PMATH

The following list of MATHLIB routines that have been included in PMATH is organized the same way as in the old MATHLIB Manual. The names given here are the MATHLIB names; refer to the [conversion chart](#) (page 14) below for the associated PMATH name(s).

Elementary Functions

None included. (Assume supplied by vendor.)

Random Number Generators

RANF	Uniform random number generator.
RANFV	Uniform random number generator (vectorized).
RNCOUNT	Count calls to RANF.
RANGET	Get RANF seed.
RANSET	Set RANF seed.
RNMUSET	Set RANF multiplier.
RLGF	Exponential random number generator.
RLGCNT	Count calls to RLGF.
RLGGET	Get RLGF seed.
RLGSET	Set RLGF seed.
RLGMSET	Set RLGF multiplier.

Maxima and Minima

MAXAF	Maximum element of array (integer).
AMAXAF	Maximum element of array (real).
MINAF	Minimum element of array (integer).
AMINAF	Minimum element of array (real).
MINMX	Minimum and maximum elements of array (integer).
AMINMX	Minimum and maximum elements of array (real).

Table Look-Up Routines

LDF	Table look-down.
LUF	Table look-up.
LUG	Table look-up with guess.

Elementary Statistical Routines

AMEANF	Mean of 1-D real array.
AMEDF	Median of 1-D real array.
STDEVF	Standard deviation of 1-D real array.
RANKS	Ranks of 1-D real array.
AMEANV	Mean vector of 2-D real array.
COVARV	Variance-covariance matrix of 2-D real array.
CORRV	Correlation matrix of 2-D real array.

Linear Algebra Routines

None included. (Assume BLAS and LINPACK supplied by vendor.)

For completeness, the following linear algebra routines were directly referenced by the former MATHLIB and are used by the S-named PMATH routines. (Obtain the D-name by replacing the initial S by D.)

BLAS :

SCOPY Copy a vector.
SDOT Dot product of two vectors.

LINPACK :

SGBFA Generate LU-factorization of a banded real matrix.
SGBSL Solve a banded linear system, given the factorization from SGBFA.
SGEFA Generate LU-factorization of a general real matrix.
SGESL Solve a general linear system, given the factorization from SGEFA.
SQRDC Generate QR-decomposition of a rectangular real matrix.
SQRSL Solve a linear least squares problem, given the SQRDC decomposition.

Root Finders

ZEROIN Zero of nonlinear function.

Interpolation and Approximation Routines

FITPOL Polynomial fit to data.
REFITP Repeated fitting after FITPOL.

Differential Equation Solvers

LSODE Ordinary differential equation solver (monotasking).
CFODE Internal routine for LSODE and NLSODE.
EWSET Internal routine for LSODE and NLSODE.
INTDY Internal routine for LSODE. (Optionally user-callable.)
PREPJ Internal routine for LSODE.
SOLSY Internal routine for LSODE.
SRCOM Internal routine for LSODE. (Optionally user-callable.)
STODE Internal routine for LSODE.
VNORM Internal routine for LSODE and NLSODE.

Miscellaneous Routines

AAAAAA Library version information.
CONSTANT Common mathematical constants.

IUMACH Standard output unit number.
RUMACH Single precision unit roundoff.

Error Procedures

XERROR Print error message.
XERRWV Print error message with value(s).
XSETUN Set error message unit number.
XSETF Set error message control flag.
IXSAV Internal routine for XERRWV, etc.

MATHLIB Routines Omitted from PMATH

These routines were omitted from PMATH because they are specific to CRAY computers or (for a few) because no significant CRAY application code called this routine in 1993.

Elementary Functions

ALOGHF Fast half-precision logarithm function.
EXPHF Fast half-precision exponential function.
SQRTHF Fast half-precision square root function.

Random Number Generators

RNFMIX Get random starting seed for RANF.
RANN Multitasking random number generator (internal seeds).
RANNINIT Automatically set all seeds for RANN.
RNNSET Set seeds for RANN.
RANGEN Multitasking random number generator (user seeds).
ISDGEN Generate seeds for RANGEN.
JMPGEN Set jump for ISDGEN.
IPOW Internal routine for JMPGEN.
IPROD Internal routine for JMPGEN.
RLGMIX Get random starting seed for RLGF.
RANWORD Generate random string.

Maxima and Minimiz

IVMAX Vectorized array maximum for CIVIC (integer).
VMAX Vectorized array maximum for CIVIC (real).
IVMIN Vectorized array minimum for CIVIC (integer).
VMIN Vectorized array minimum for CIVIC (real).

Differential Equation Solvers

NLSODE Ordinary differential equation solver (multitasking).
INTY Internal routine for NLSODE.
INTYD Internal routine for NLSODE.
JPREP Internal routine for NLSODE.
SYSOL Internal routine for NLSODE.
SZRCM Internal routine for NLSODE.
TSODE Internal routine for NLSODE.

Error Procedures

LERRW Multitasking version of XERRWV.
LERIN Initialize LERRW.

Conversion of MATHLIB to PMATH Names

This chart lists the names of former MATHLIB routines and the corresponding single- and double-precision names of PMATH routines. In cases where the routine processes only integer data or where the S- and D-versions are dependent (see Principle 2.1 above), there is only one PMATH name and it appears only in the righthand column. Each PMATH routine (version) has one descriptive prolog, reprinted later in this manual (arranged alphabetically by routine name) and linked to the entry in the righthand column for easy online access (you get the S-name version if there is one).

MATHLIB (UNICOS) name	S-name	-PMATH- D-name	REAL*8 name
RANF	SRANF	DRANF	<u>RANF8</u>
RANFV	SRANFV	DRANFV	<u>RANFV8</u>
RNCOUNT	---	---	<u>RNFCNT</u>
RANGET	---	---	<u>RNSGET</u>
RANSET	---	---	<u>RNSSET</u>
RNMUSET	---	---	<u>RNMSET</u>
RLGF	SRLGF	DRLGF	<u>RLGF8</u>
RLGCNT	---	---	<u>RLFCNT</u>
RLGGET	---	---	<u>RLSGET</u>
RLGSET	---	---	<u>RLSSET</u>
RLGMSET	---	---	<u>RLMSET</u>
MAXAF	---	---	<u>IMAXAF</u>
AMAXAF	SMAXAF	DMAXAF	<u>AMAXF8</u>
MINAF	---	---	<u>IMINAF</u>
AMINAF	SMINAF	DMINAF	<u>AMINF8</u>
MINMX	---	---	<u>IMINMX</u>
AMINMX	SMINMX	DMINMX	<u>AMNMX8</u>
LDF	LDFS	LDFD	<u>LDF8</u>
LUF	LUFS	LUFD	<u>LUF8</u>
LUG	LUGS	LUGD	<u>LUG8</u>
AMEANF	SMEANF	DMEANF	<u>AMEAN8</u>
AMEDF	SMEDF	DMEDF	<u>AMED8</u>
STDEVF	SSTDEV	DSTDEV	<u>STDEV8</u>
RANKS	SRANKS	DRANKS	<u>RANKS8</u>
AMEANV	SMEANV	DMEANV	<u>MEANV8</u>
COVARV	SCOVAR	DCOVAR	<u>COVAR8</u>
CORRV	SCORRV	DCORRV	<u>CORRV8</u>
ZEROIN	SZERO	DZERO	<u>ZERO8</u>
FITPOL	SFITPO	DFITPO	<u>FITPO8</u>
REFITP	SREFIT	DREFIT	<u>REFIT8</u>
LSODE	SLSODE	DLSODE	<u>LSODE8</u>
AAAAAA	---	---	<u>AAAAAA</u>
CONSTANT	SCONST	DCONST	<u>CONST8</u>
IUMACH	---	---	<u>IUMACH</u>
RUMACH	RUMACH	DUMACH	<u>UMACH8</u>

XERROR	---	---	<u>XERROR</u>
XERRWV	XERRWV	XERRWD	<u>XERRWV</u>
XSETF	---	---	<u>XSETF</u>
XSETUN	---	---	<u>XSETUN</u>
---	---	---	<u>CV16TO64</u>
---	---	---	<u>CV64TO16</u>

PMATH Routines by Category with PMATH Names

PMATH Routines Grouped by Function

The following is a categorized list of the contents of PMATH. If more than one name is given, the first is the S-name, the second is the D-name, and the third is the REAL*8 name. (Here "S" refers to single precision, "D" to double precision.) Only user-callable routines have REAL*8 names. If only one name is given, the routine is either typeless or does not process floating-point data.

There are no REAL*8-named routines in the library (except for RANF8 and RLGF8). Each name is translated to an equivalent S- or D-name, depending on the precision of the platform.

The CV-routines are new to PMATH, and are provided to assist in correctly moving 48-bit integers (like random number generator seeds) to or between 32-bit platforms.

Elementary Functions

None included. (Assume supplied by vendor.)

Random Number Generators

- SRANF (page 132)/DRANF (page 61)/RANF8 (page 82) Uniform random number generator.
- SRANFV (page 133)/DRANFV (page 62)/RANFV8 Uniform random number generator (vectorized).
- RNFCNT (page 90) Count calls to RANF.
- RNSGET (page 93) Get RANF seed.
- RNSSET (page 94) Set RANF seed.
- RNMSET (page 91) Set RANF multiplier.
- SRLGF (page 137)/DRLGF (page 66)/RLGF8 Exponential random number generator.
- RLFCNT (page 83) Count calls to RLGF.
- RLSGET (page 87) Get RLGF seed.
- RLSSET (page 88) Set RLGF seed.
- RLMSET (page 85) Set RLGF multiplier.

Maxima and Minima

- IMAXAF (page 72) Maximum element of array (integer).
- SMAXAF (page 126)/DMAXAF (page 55)/AMAXF8 Maximum element of array (real).
- IMINAF (page 73) Minimum element of array (integer).
- SMINAF (page 130)/DMINAF (page 59)/AMINF8 Minimum element of array (real).
- IMINMX (page 74) Minimum and maximum of array (integer).
- SMINMX (page 131)/DMINMX (page 60)/AMNMX8 Minimum and maximum of array (real).

Table Look-Up Routines

- LDFS (page 77)/LDFD (page 76)/LDF8 Table look-down.
- LUFS (page 79)/LUFD (page 78)/LUF8 Table look-up.
- LUGS (page 81)/LUGD (page 80)/LUG8 Table look-up with guess.

Elementary Statistical Routines

- SMEANF (page 127)/DMEANF (page 56)/AMEAN8 Mean of 1-D real array.
- SMEDF (page 129)/DMEDF (page 58)/AMED8 Median of 1-D real array.
- SSTDEV (page 139)/DSTDEV (page 68)/STDEV8 Standard deviation of 1-D real array.
- SRANKS (page 134)/DRANKS (page 63)/RANKS8 Ranks of 1-D real array.
- SMEANV (page 128)/DMEANV (page 57)/MEANV8 Mean vector of 2-D real array.
- SCOVAR (page 99)/DCOVAR (page 28)/COVAR8 Variance-covar. matrix of 2-D real array.
- SCORRV (page 98)/DCORRV (page 27)/CORRV8 Correlation matrix of 2-D real array.

Linear Algebra Routines

None included. (Assume BLAS and LINPACK supplied by vendor.)

Root Finders

- SZERO (page 140)/DZERO (page 70)/ZERO8 Zero of nonlinear function.

Interpolation and Approximation Routines

- SFITPO (page 101)/DFITPO (page 30)/FITPO8 Polynomial fit to data.
- SREFIT (page 135)/DREFIT (page 64)/REFIT8 Repeated fitting after FITPOL.

Differential Equation Solvers

- LSODE (page 106)/DLSODE (page 35)/LSODE8 Ordinary differential equation solver (monotasking).
- SCFODE/DCFODE Internal routine for LSODE.
- SEWSET/DEWSET Internal routine for LSODE.
- SINTDY/DINTDY/INTDY8 Internal routine for LSODE. (Optionally user-callable.)
- SPREPJ/DPREPJ Internal routine for LSODE.
- SOLSY/DSOLSY Internal routine for LSODE.
- SSRCOM (page 138)/DSRCOM (page 67)/SRCOM8 Internal routine for LSODE. (Optionally user-callable.)
- SSTODE/DSTODE Internal routine for LSODE.
- SVNORM/DVNORM Internal routine for LSODE.

Miscellaneous Routines

- AAAAAA (page 22) Library version information.
- SCONST (page 96)/DCONST (page 25)/CONST8 Common mathematical constants.
- IUMACH (page 75) Standard output unit number.
- RUMACH (page 95)/DUMACH (page 69)/UMACH8 Single precision unit roundoff.
- CV16TO64 (page 23) Convert from 16-bit (2-byte) to 64-bit (8-byte) format.
- CV64TO16 (page 24) Convert from 64-bit (8-byte) to 16-bit (2-byte) format.

Error Procedure

- XERROR (page 142) Print error message.
- XERRWV (page 144)/XERRWD (page 143) Print error message with value(s).
- XSETUN (page 146) Set error message unit number.
- XSETF (page 145) Set error message control flag.
- IXSAV Internal routine for XERRWV, etc.

Routines New to PMATH Explained

This section contains descriptions of PMATH routines that were not in MATHLIB.

Vectorized RANF

(See [Principle 2.2](#) (page 7), above.) The following Fortran interface has been defined for the portable equivalent of the compiler-generated RANFV.

Usage:

```
INTEGER N REAL RANOUT(n) or
DOUBLE PRECISION RANOUT(n) or
REAL*8 RANOUT(n)
```

```
CALL SRANFV(N,RANOUT) or
CALL DRANFV(N,RANOUT) or
CALL RANFV8(N,RANOUT)
```

Arguments:

```
N      (in)      Number of random numbers to be generated.
RANOUT (out)     Vector of N random numbers between 0 and 1.
                        The actual dimension of RANOUT must satisfy n>=N.
```

Description:

SRANFV/DRANFV/RANFV8 generates pseudorandom numbers lying strictly between 0 and 1. The above call is equivalent to the loop:

```
DO 10 I=1,N
  RANOUT(I) = RANF( )
10 CONTINUE
```

where RANF is SRANF/DRANF/RANF8 for SRANFV/DRANFV/RANFV8, respectively. Note that SRANFV/DRANFV/RANFV8 may be significantly faster for large N. (The actual timing is likely to be platform-dependent.) The current implementation merely contains a DO-loop, as per the specification.

Portable Seed-Passing Module

(See [Principle 2.5](#) (page 7), above.) A pair of routines, C V 16T O 64 and C V 64T O 16, has been developed to facilitate moving random number seeds (which are 48-bit integers) to or between 32-bit platforms. The output from RNSGET (RLSGET) can be unpacked via CV64TO16 and written for export with a 3Z4 format. The integers may then be read with this same format, repacked via CV16TO64, and used as the argument to RNSSET (RLSSET). Multipliers for RNMSET (RLMSET) can also be constructed via CV16TO64.

It is intended that these routines be used as follows:

```
(for RANF family)      (for RLGF family)
REAL*8 RNSGET, SEED    or REAL*8 RLSGET, SEED
INTEGER ISEED(3)
```

```

[ Compute for a while. ]
SEED = RNSGET ( )           or  SEED = RLSGET ( )
CALL CV64TO16 (SEED, ISEED)
[ Write ISEED to dumpfile via FORMAT(3Z4). ]

[ In restart code, read ISEED via FORMAT(3Z4). ]
CALL CV16TO64 (ISEED, SEED)
CALL RNSSET (SEED)           or  CALL RLSSET (SEED)
[ Continue computation. ]

```

The Fortran interface for CV16TO64 is as follows:

Usage:

```

INTEGER IN16(3)
REAL*8  OUT64
CALL CV16TO64 (IN16, OUT64)

```

Arguments:

```

IN16  (in)   An array containing three 16-bit integers.
OUT64 (out)   The result of packing these into the low-order
               bits of a 64-bit word.
               Its leftmost 16 bits will be zero;
               its rightmost 16 bits will be from IN16(3).

```

Description:

```

OUT64 = +-----+
        |      0      | IN16(1) | IN16(2) | IN16(3) |
        +-----+

```

The Fortran interface for CV64TO16 is as follows:

Usage:

```

REAL*8  IN64
INTEGER OUT16(3)
CALL CV64TO16 (IN64, OUT16)

```

Arguments:

```

IN64  (in)   The 64-bit quantity to be converted.
OUT16 (out)   An array of 16-bit quantities containing the
               rightmost 48 bits from IN64.
               The rightmost 16 bits will be in OUT16(3).

```

Description:

```

IN64 = +-----+
        | ignored | OUT16(1) | OUT16(2) | OUT16(3) |
        +-----+

```

This C module is contained in `pmath_cnv.c`, which requires the header files `pm_params.h` and `pm_cnvset.h` to set up correct Fortran binding. Because of problems with C on the CRAYs, the former CRAY version had been written in LRLTRAN (CIVIC) code.

PMATH Routine Descriptions

The subsections of this section each contain a descriptive prolog for one PMATH subroutine, arranged in alphabetical order by routine name. Included is the calling sequence and other usage details. To see a task-oriented overview of the PMATH library, consult the earlier section called PMATH Routines Grouped by Function (page 16).

AAAAAA

```
      SUBROUTINE AAAAAA (VER)
***BEGIN PROLOGUE  AAAAAA
***PURPOSE  LLNL Portable Mathematical Library disclaimer and version.
***LIBRARY   PMATH
***CATEGORY  Z
***TYPE      ALL (AAAAAA-A)
***KEYWORDS  DISCLAIMER, DOCUMENTATION, VERSION
***AUTHOR   LC Mathematical Software Service
***DESCRIPTION
```

PMATH is a portable version of MATHLIB, the standard mathematical library for LLNL Cray's and earlier machines. These routines are distributed exclusively for use in support of LLNL programs. Check with the LLNL Code Release Center or the LC Client Services HotLine, (510)422-4531, before moving this source code to a non-LLNL system.

```
+-----+
+               * * * * * Notice * * * * *               +
+
+ This material was prepared as an account of work sponsored +
+ by the United States government. Neither the United       +
+ States government nor any of their employees, nor any of   +
+ their contractors, subcontractors, or their employees,    +
+ makes any warranty, expressed or implied, or assumes any  +
+ legal liability or responsibility for the accuracy,        +
+ completeness or usefulness of any information, apparatus, +
+ product or process disclosed, or represents that its use  +
+ would not infringe privately-owned rights.                +
+-----+
```

*Usage:

```
      CHARACTER*24 VER
```

```
      CALL AAAAAA (VER)
```

*Arguments:

VER:OUT will contain the version number of PMATH.

*Description:

This routine contains the PMATH disclaimer and can be used to return the library version number.

```
***END PROLOGUE  AAAAAA
```

CV16TO64

```
***BEGIN PROLOGUE  CV16TO64
***PURPOSE  Convert from an array of 16-bit quantities to a
             64-bit word.
***LIBRARY   PMATH
***CATEGORY  N2
***TYPE      ALL (CV16TO64-A)
***KEYWORDS  CONVERSION, PACKING
***AUTHOR    Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
```

***Usage:**

```
      INTEGER IN16(3)
      REAL*8  OUT64
      CALL CV16TO64 (IN16, OUT64)
```

***Arguments:**

IN16 :IN An array containing three 16-bit integers.

OUT64:OUT The result of packing these into the low-order bits of
 a 64-bit word. Its leftmost 16 bits will be zero; its
 rightmost 16 bits will be from IN16(3).

***Description:**

```
      OUT64  =  +-----+
                 |      0      | IN16(1) | IN16(2) | IN16(3) |
                 +-----+
```

CV16TO64 and CV64TO16 were developed to facilitate moving random number seeds (which are 48-bit integers) to or between 32-bit platforms. The output from RNSGET (RLSGET) can be unpacked via CV64TO16 and written for export with a 3Z4 format. The integers may then be read with this same format, repacked via CV16TO64, and used as the argument to RNSSET (RLSSET). Multipliers for RNMSET (RLMSET) can also be constructed via CV16TO64.

***See also:**

See CV64TO16 description for an example.

***Portability:**

This C routine is contained in pmath_cnv.c, which requires header files pm_params.h and pm_cnvset.h to set up correct Fortran binding.

```
***END PROLOGUE  CV16TO64
```

CV64TO16

```
***BEGIN PROLOGUE  CV64TO16
***PURPOSE  Convert from a 64-bit word to an array of 16-bit
            quantities.
***LIBRARY   PMATH
***CATEGORY  N2
***TYPE      ALL (CV64TO16-A)
***KEYWORDS  CONVERSION, UNPACKING
***AUTHOR    Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
```

*Usage:

```
      REAL*8  IN64
      INTEGER OUT16(3)
      CALL CV64TO16 (IN64, OUT16)
```

*Arguments:

IN64 :IN The 64-bit quantity to be converted.

OUT16:OUT An array of 16-bit quantities containing the rightmost
 48 bits from IN64. The rightmost 16 bits will be in
 OUT16(3).

*Description:

```
      IN64  =  +-----+
                | ignored | OUT16(1) | OUT16(2) | OUT16(3) |
                +-----+
```

CV16TO64 and CV64TO16 were developed to facilitate moving random number seeds (which are 48-bit integers) to or between 32-bit platforms. The output from RNSGET (RLSGET) can be unpacked via CV64TO16 and written for export with a 3Z4 format. The integers may then be read with this same format, repacked via CV16TO64, and used as the argument to RNSSET (RLSSET).

*Example:

```
      (for RANF family)                (for RLGF family)
      REAL*8  RNSGET, SEED              or      REAL*8  RLSGET, SEED
      INTEGER ISEED(3)
      < Compute for a while. >
      SEED = RNSGET ( )                  or      SEED = RLSGET ( )
      CALL CV64TO16 (SEED, ISEED)
      < Write ISEED to dumpfile via FORMAT (3Z4). >
      .
      .
      .
      < In restart code, read ISEED via FORMAT (3Z4). >
      CALL CV16TO64 (ISEED, SEED)
      CALL RNSSET (SEED)                  or      CALL RLSSET (SEED)
      < Continue computation. >
```

*Portability:

This C routine is contained in pmath_cnv.c, which requires header files pm_params.h and pm_cnvset.h to set up correct Fortran binding.

```
***END PROLOGUE  CV64TO16
```


DCONST

```
DOUBLE PRECISION FUNCTION DCONST (NAME)
***BEGIN PROLOGUE  DCONST
***PURPOSE  Provides values for common mathematical constants.
***LIBRARY   PMATH
***CATEGORY  R1
***TYPE      DOUBLE PRECISION (SCONST-S, DCONST-D, CONST8-8)
***KEYWORDS  CONSTANTS, PI, TWOPI, PI180, PI3, TWOPI3, FOURPI3, UROUND,
              ONE3, ONE27
***AUTHOR    Basinger, R.C., (LLNL/CMRD)
              Currently responsible:
              Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine CONSTANT.)
*Usage:
    CHARACTER*n  NAME
    DOUBLE PRECISION  VALUE, DCONST

    NAME = 'name'
    VALUE = DCONST (NAME)
or
    VALUE = DCONST ('name')
```

*Arguments:

NAME :IN Name of the desired constant. Valid names and their meanings are:

I	Name	Value	Meaning
1	'pi'	pi	PI = 4.0*ATAN(1.0)
2	'twopi'	2pi	2.0*PI
3	'pi180'	pi/180	PI/180.0
4	'pi3'	pi/3	PI/3.0
5	'twopi3'	2pi/3	2.0*PI/3.0
6	'fourpi3'	4pi/3	4.0*PI/3.0
7	'uround'	unit roundoff	The smallest positive floating- point number such that 1.0 + 'uround' .NE. 1.0
8	'one3'	1/3	1.0/3.0
9	'one27'	1/27	1.0/27.0

Here "pi" in the Value column represents the Greek letter pi, the standard notation for the ratio of the circumference to the diameter of a circle.

The name of the constant may be given in either upper or lower case (but not mixed case).

*Function Return Values:

VALUE : the value of the named constant.

*Description:

DCONST provides values for commonly used mathematical constants. This provides a machine-independent way to obtain correct values for these constants.

*Accuracy:

All values except for element 7 are data-loaded with 32-digit decimal constants generated using Macsyma. We rely on the compiler generating correctly rounded machine values from them. SCONST('uround') is obtained from RUMACH.

*Cautions:

The present version terminates with a STOP statement if NAME is not a valid name.

***REFERENCES (NONE)

***ROUTINES CALLED DUMACH

***REVISION HISTORY (YYMMDD)

820514 DATE WRITTEN

(The above is the date found in the source code. It may be an underestimate of the age of this routine.)

890224 Added SLATEC/LDOC prologue. (FNF)

890301 Made changes to comments per feedback from Tok. (FNF)

890301 Replaced double quote (") as string delimiter in DATA statements with the ANSI standard single quote ('). (FNF)

900627 Changed hexadecimal constants from CIVIC to CFT77 form. (FNF)

920313 Made minor cosmetic changes and changed DATA-loaded value of N to the actual number of available constants. (FNF)

920316 Modified to recognize either upper or lower case names.

Removed the common blocks in the process. (FNF)

920319 Updated with prologue edited 891025 by G. Shaw for manual.

930823 1. Replaced calls to BASELIB routine ZVSEEK with a loop.

2. Rearranged DATA statements to facilitate subsequent changes. (FNF)

930824 Changed names from INTEGER to the more standard CHARACTER type. (FNF)

930826 Eliminated distinction between N, the number of constants, and the dimensions of the arrays. (FNF)

930830 Added decimal values, surrounded by suitable comments, for all constants except machine precision. (FNF)

***END PROLOGUE DCONST

DCORRV

```
SUBROUTINE DCORRV (VCV, M, WK)
***BEGIN PROLOGUE  DCORRV
***PURPOSE  Calculate the correlation matrix from the variance-
             covariance matrix.
***LIBRARY   PMATH
***CATEGORY  L1B
***TYPE      DOUBLE PRECISION (SCORRV-S, DCORRV-D, CORRV8-8)
***KEYWORDS  ELEMENTARY STATISTICS, CORRELATION MATRIX
***AUTHOR    Unknown, Name (LLNL/USD/NMG)
             Durst, Mark J. (LLNL/CMRD/SPG)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
             (Portable version of Cray MATHLIB routine CORRV.)
*Usage:
      INTEGER  M
      PARAMETER (nvcv = (M*(M+1))/2)
      DOUBLE PRECISION  VCV(nvcv), WK(M)
      CALL DCORRV (VCV, M, WK)

*Arguments:
      VCV:INOUT   Input:  Array of order M(M + 1)/2 containing the
                    variance-covariance matrix in symmetric storage mode.
                    Output: Array containing the correlation matrix in
                    symmetric storage mode.
      M :IN       Number of variables for which correlations are
                    calculated.
      WK :WORK     Work array of order M.

*Description:
      DCORRV calculates the correlation matrix from the variance-
      covariance matrix stored in VCV in symmetric storage mode.  The
      correlation matrix will replace VCV on return.

      "Symmetric storage mode" means (S is taken to be the full matrix):

      VCV(k) = S(i,j),    k = (i(i - 1))/2 + j, i = 1,...,M, j <= i

*See Also:
      DCORRV can be used in conjunction with DCOVAR to obtain both the
      variance-covariance and correlation matrices.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
              (The above is the date of LCSD-442,Rev.1 and is undoubtedly
              a significant underestimate of the age of this routine.)
      890223  Added SLATEC/LDOC prologue. (FNF)
***END PROLOGUE  DCORRV
```

DCOVAR

```
SUBROUTINE DCOVAR (A, N, M, IND, VCV, SD, WK)
***BEGIN PROLOGUE  DCOVAR
***PURPOSE  Variance-covariance or correlation matrix of a
             two-dimensional real array.
             Calculates the standard deviations and the variance-
             covariance or correlation matrix for N observations on
             each of M variables.
***LIBRARY    PMATH
***CATEGORY   L1B
***TYPE       DOUBLE PRECISION (SCOVAR-S, DCOVAR-D, COVAR8-8)
***KEYWORDS   ELEMENTARY STATISTICS, STANDARD DEVIATION, VECTOR,
             VARIANCE-COVARIANCE MATRIX, CORRELATION MATRIX
***AUTHOR     Unknown, Name (LLNL/USD/NMG)
             Durst, Mark J. (LLNL/CMRD/SPG)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
             (Portable version of Cray MATHLIB routine COVARV.)
*Usage:
      INTEGER  N, M, IND
      PARAMETER (nvcv = (M*(M+1))/2)
      DOUBLE PRECISION  A(N,M), VCV(nvcv), SD(M), WK(M)
      CALL DCOVAR (A, N, M, IND, VCV, SD, WK)

*Arguments:
      A  :IN      N by M array of N observations on M variables.
      N  :IN      Row dimension of A.
      M  :IN      Column dimension of A.
      IND:IN      Job-control flag:
                   0      Return the variance-covariances.
                   non-0   Return correlations.
      VCV:OUT     Array of order M(M+1)/2 containing either the
                   variance-covariances or correlations in symmetric
                   storage mode, depending on the value of IND.
      SD :OUT     Array of order M containing the standard deviations.
      WK:WORK     Work array of order M.

*Description:
      DCOVAR calculates the standard deviation in SD and the
      variance-covariance matrix in VCV in symmetric storage mode.  If
      IND does not equal 0, it then calls DCORRV to calculate the
      correlation matrix from the variance-covariance matrix.

      "Symmetric storage mode" means (S is taken to be the full matrix):

      VCV(k) = S(i,j), k = (1(i-1))/2 + j, i = 1,...,M, j <=1

*See Also:
      If both the variance-covariance matrix and the correlation matrix
      are required, first call DCOVAR with IND = 0. Then copy VCV into
      the desired array for the correlation matrix and call DCORRV.

***REFERENCES  (NONE)
***ROUTINES CALLED  DCORRV
***REVISION HISTORY  (YYMMDD)
```

```

830812  DATE WRITTEN
        (The above is the date of LCSD-442,Rev.1 and is undoubtedly
        a significant underestimate of the age of this routine.)
890223  Added SLATEC/LDOC prologue. (FNF)
890518  Modified sequence numbers to fit in columns 73-80. (FNF)
890518  1. Replaced expr**.5 with sqrt(expr)--one occurrence. (FNF)
        2. Corrected dimension for array VCV. (FNF)
890519  Eliminated redundant variable ink.
920319  Updated with prologue edited 891025 by G. Shaw for manual.
930706  Corrected C***CATEGORY line. (FNF)
930930  Converted old UNICOS names to S- or I-names. (DBP)
931005  Corrected list of equivalent routines, made sure that all
        variables are declared, and improved comments. (FNF)
931018  Produced double precision version. (DBP)
931026  Minor changes to reduce single/double differences. (FNF)
931029  Changed back to generic intrinsics. (FNF)
940421  Improved purpose. (FNF)
940727  Added preprocessor directives for REAL*8 entries. (FNF)
951010  Corrected LIBRARY line. (FNF)
***END PROLOGUE  DCOVAR

```

DFITPO

```
SUBROUTINE DFITPO (XDATA, YDATA, NDATA, NTERMS, WEIGHT, COEFF,  
+                RSD2, WORK, JOB, IERR)  
***BEGIN PROLOGUE  DFITPO  
***PURPOSE  Fit a polynomial to given data.  
            Finds the polynomial that is the best least-squares  
            fit to a given set of data points.  
***LIBRARY    PMATH  
***CATEGORY   K1A1A2, L8B1B1  
***TYPE       DOUBLE PRECISION (SFITPO-S, DFITPO-D, FITPO8-8)  
***KEYWORDS   POLYNOMIAL FITTING, LEAST SQUARES  
***AUTHOR    Painter, Jeffrey F., (LLNL/CMRD)  
            Currently responsible:  
            Fritsch, Fred N., (LLNL/LC/MSS)  
***DESCRIPTION  
            (Portable version of Cray MATHLIB routine FITPOL.)  
*Usage:  
    INTEGER  NDATA, NTERMS, JOB, IERR  
    PARAMETER (NWORK = (NDATA+1)*(NTERMS+1) )  
    DOUBLE PRECISION  XDATA(NDATA), YDATA(NDATA), WEIGHT(NDATA),  
    *                  COEFF(NTERMS), RSD2, WORK(NWORK)  
  
    CALL DFITPO (XDATA, YDATA, NDATA, NTERMS, WEIGHT, COEFF,  
    *            RSD2, WORK, JOB, IERR)  
  
*Arguments:  
    In the following, the data points are  
    (x(i),y(i)) = ( XDATA(i), YDATA(i) ), i=1,...,NDATA.  
  
XDATA :IN    Array of values of the independent variable, x, among  
            which there must be at least NTERMS different values.  
            Its dimension is NDATA.  
  
YDATA :IN    Array of corresponding values of the dependent  
            variable, y.  Its dimension is NDATA.  
  
NDATA :IN    The number of data points to be fit.  
  
NTERMS:IN    The number of terms in the polynomial (i.e., DFITPO  
            is to determine a polynomial of degree NTERMS - 1).  
            If NTERMS > NDATA, the result will be the coefficients  
            of an interpolating polynomial of degree NDATA-1, and  
            COEFF(j) = 0 for j > NDATA.  
  
WEIGHT:IN    Optional weight array.  
  
            If WEIGHT(1) is equal to zero, DFITPO will choose  
            COEFF to minimize the sum of the squares of the  
            residuals.  In this case, WEIGHT need not be  
            dimensioned and can, indeed, be the literal 0.D0.  
  
            Otherwise, WEIGHT must be an array of dimension  
            NDATA, with WEIGHT(1) nonzero, and DFITPO will choose  
            COEFF to minimize the sum of the squares of the  
            weighted residuals,  
             $R(i) = \text{WEIGHT}(i) * (y(i) - p(x(i)))$ , i=1,2,...,NDATA.  
            (See Description, below, for definition of p(x).)
```

COEFF:OUT Array containing the NTERMS coefficients of the polynomial. COEFF(j) is the coefficient of $x^{(j-1)}$.

RSD2 :OUT Sum of the squares of the (weighted) residuals corresponding to COEFF.

WORK :WORK Array used primarily for internal computations. NWORK, its dimension, must be at least $(\text{NDATA}+1)*(\text{NTERMS}+1)$. If JOB is nonzero, the first NDATA words of WORK will contain the residuals (or weighted residuals, if the weighting option was chosen) on return:
 $\text{WORK}(i) = R(i), i = 1, 2, \dots, \text{NDATA}.$
 Note that if DREFIT is to be used for subsequent fits, WORK must not be modified in any way.

JOB :IN Residuals-computation flag:
 non-0 Residuals are computed and output in WORK.
 0 Residuals are not completely computed, although RSD2 is computed. (This option will be more efficient if the $R(i)$ are not required.)

IERR :OUT Error flag. On normal termination, IERR = 0.

Warning error: IERR <= -4
 In this case the problem looks poorly conditioned, so that all components of COEFF may be inaccurate. $10*(-\text{IERR})$ will be a lower bound for the condition number, and COEFF will be computed anyway. (See "Accuracy" below for details.)

Fatal error:
 DQRSL returned INFO=IERR: $0 < \text{IERR} \leq \text{NTERMS}$
 A singular matrix has been detected. This may be due to too many values of XDATA(i) exactly equal or too many weights equal to zero.
 COEFF has not been computed in this case.

***Description:**

DFITPO finds the polynomial that is the best least-squares fit to a given set of data points

$$(x(i), y(i)) = (XDATA(i), YDATA(i)), \quad i = 1, 2, \dots, \text{NDATA}.$$

It finds coefficients COEFF(1), ..., COEFF(NTERMS) of the polynomial

$$y = p(x) = \text{COEFF}(1) + \text{COEFF}(2)*x + \text{COEFF}(3)*x^2 + \dots + \text{COEFF}(\text{NTERMS})*x^{(\text{NTERMS}-1)},$$

which minimize the sum of the squares of the residuals

$$R(i) = y(i) - p(x(i)), \quad i = 1, 2, \dots, \text{NDATA}.$$

As an option, the residuals may be weighted, as noted above.

If the range of x-values is far from zero, DFITPO may introduce extra inaccuracies in the results, especially in lower-order coefficients. A way to get better results is to choose a typical value of x, say x_0 , and define

```
xnew(i) = x(i) - x0, i = 1, 2, ..., NDATA .
```

Then instead of

```
CALL DFITPO (x, ...)
```

use

```
CALL DFITPO (xnew, ...) .
```

The result will be coefficients for the polynomial

```
y = p(xnew) = p(x-x0) .
```

Let A denote the matrix whose i-th row is

```
( 1   XDATA(i)   XDATA(i)**2   ....   XDATA(i)**(NTERMS-1) )
```

(This row is multiplied by WEIGHT(i) if the weighting option has been chosen.) A is called the least-squares matrix. The solution to the least-squares problem is found by way of a QR decomposition of A, without pivoting, using LINPACK routines DQRDC and DQRSL.

The covariance matrix of COEFF can be estimated after a call of DFITPO. If all the data points $y = YDATA(i)$ have the same variance $v(y)$, then the covariance matrix is $v(y)$ times the inverse of the product of A-transpose (denoted A^t) and A:

```
cov = v(y) * inv(At*A) ,
```

An estimate of $v(y)$ is $RSD2/(NDATA - NTERMS)$. The following call of a LINPACK subroutine (Ref. 1) will compute $inv(A^t A)$:

```
CALL DPODI (WORK(2+NDATA), NDATA, NTERMS, DUMMY, 1)
```

where WORK, NDATA, and NTERMS are the same variables as in DFITPO, WORK has not been disturbed since the last DFITPO call, and DUMMY is not referenced. Only WORK is changed. For $i \leq j$, DPODI puts the (i,j)th element of $inv(A^t A)$ (which equals the (j,i)th element) into $WORK(i+j*NDATA+1)$. CAUTION: Since this changes WORK, DREFIT cannot be called after such a call of DPODI.

Sometimes an expression involving $inv(A^t A)$ can be evaluated without computing the inverse; if so, and if NTERMS is large, it will be cheaper not to compute the inverse. An equation of the form

```
(At*A) * w = b
```

can best be solved for w by the following call of a LINPACK routine (Ref. 1):

```
CALL DPOSL (WORK(2+NDATA), NDATA, NTERMS, BW)
```

where WORK, NDATA, and NTERMS are input variables, undisturbed since the last DFITPO call, and BW is a real vector of dimension NTERMS. On input, BW is b, and on output, it is w. Since DPOSL does not change WORK, you may call DREFIT or DPOSL after calling

DPOSL.

***Examples:**

See the DREFIT writeup for a sample call of DFITPO.

The following sample code is a faster way to evaluate the polynomial $Y = p(X)$ than the most straightforward approach.

```
      Y = COEFF(NTERMS)
      DO 10 J = 1, (NTERMS - 1)
10    Y = X*Y + COEFF(NTERMS - J)
```

***Accuracy:**

DFITPO finds a lower bound for the condition number K of the problem. This number is relevant because DFITPO will introduce an error in each $\text{COEFF}(j)$ ($j = 1, 2, \dots, \text{NTERMS}$) that is roughly proportional to K times the largest of these coefficients (larger if there are large values of x in the data). If the condition-number estimate is over 10,000, then the error flag IERR will be set to a negative number so that K is greater than $10^{**}(|\text{IERR}|)$. It is unlikely that K will be any larger than $10^{**}(|\text{IERR}| + 2)$. As a rule of thumb, this means that the largest of the coefficients may have lost about $|\text{IERR}| + 2$ digits of accuracy. The same absolute error estimate applies to all of the coefficients; thus, if $\text{COEFF}(j)$ is smaller than the largest coefficient by a factor of $10^{**}n$, it will have lost $|\text{IERR}| + 2 + n$ digits of accuracy. If some values of x are large and if NTERMS is large, then lower-order coefficients will be less accurate. For details, see Ref. 1, pp. I.8-I.11 and 9.4-9.5, and Ref. 2, pp. 28-35.

The above discussion applies to the mathematical fitting problem; of course there may be other inaccuracies from the input data. Furthermore, the polynomial computed when $\text{IERR} < 0$ may be perfectly acceptable if all one needs is a function that produces small residuals.

***Cautions:**

DFITPO assumes $1 \leq \text{NTERMS}$, NDATA . This is not checked. See description of NTERMS for behavior when $\text{NTERMS} > \text{NDATA}$.

This is a simple program for simple problems. It is not recommended for large problems.

***Portability:**

This routine calls the LINPACK routines DQRDC and DQRSL, and BLAS (Basic Linear Algebra Subprograms) DDOT.

The declaration `REAL WORK(NDATA,*)` is used to cause the compiler to generate suitable subscript arithmetic for the NDATA by NTERMS least-squares matrix stored starting at element `WORK(2,2) = WORK(NDATA+2)`. Some compilers may object to the fact that $(I+1) > \text{NDATA}$ when $I = \text{NDATA}$ in loops 10, 30 and 50.

***ROUTINES CALLED DDOT, DQRDC, DQRSL

***REVISION HISTORY (YYMMDD)

800301 DATE WRITTEN

890419 Added SLATEC/LDOC prologue. (FNF)

890424 Corrected DATE WRITTEN. (FNF)

890518 Modified sequence numbers to fit in columns 73-80. (FNF)

920319 Updated with prologue edited 891025 by G. Shaw for manual.
920331 Reformatted references section. (FNF)
930706 Corrected C***CATEGORY line. (FNF)
930930 Converted old UNICOS names to S- or I-names. (DBP)
931005 Augmented list of equivalent routines, made sure that all
variables are declared, and improved comments. (FNF)
931018 Produced double precision version. (DBP)
931026 Minor changes to reduce single/double differences. (FNF)
931029 Changed back to generic intrinsics. (FNF)
***END PROLOGUE DFITPO

DLSODE

```
      SUBROUTINE DLSODE (F, NEQ, Y, T, TOUT, ITOL, RTOL, ATOL, ITASK,  
+                      ISTATE, IOPT, RWORK, LRW, IWORK, LIW, JAC, MF)  
***BEGIN PROLOGUE  DLSODE  
***PURPOSE  Livermore solver for ordinary differential equations.  
            Solves the initial-value problem for stiff or nonstiff  
            systems of first-order ODE's,  
             $dy/dt = f(t,y)$ , or, in component form,  
             $dy(i)/dt = f(i) = f(i,t,y(1),y(2),\dots,y(N))$ ,  $i=1,\dots,N$ .  
***LIBRARY  PMATH (ODEPACK)  
***CATEGORY  I1A1B, I1A2  
***TYPE      DOUBLE PRECISION (SLSODE-S, DLSODE-D, LODE8-8)  
***KEYWORDS  ORDINARY DIFFERENTIAL EQUATIONS, INITIAL VALUE PROBLEM,  
            STIFF, NONSTIFF  
***AUTHOR  Hindmarsh, Alan C., (LLNL)  
            Center for Computational Sciences and Engrg., L-316  
            Lawrence Livermore National Laboratory  
            Livermore, CA 94550.  
***DESCRIPTION  
            (Portable version of Cray MATHLIB routine LODE.)
```

NOTE: The DLSODE solver is not re-entrant, and so is usable on the Cray multi-processor machines only if it is not used in a multi-tasking environment.
If re-entrancy is required, use NLSODE instead.

The formats of the DLSODE and NLSODE writeups differ from those of the other MATHLIB routines.

The "Usage" and "Arguments" sections treat only a subset of available options, in condensed fashion. The options covered and the information supplied will support most standard uses of DLSODE.

For more sophisticated uses, full details on all options are given in the concluding section, headed "Long Description." A synopsis of the DLSODE Long Description is provided at the beginning of that section; general topics covered are:

- Elements of the call sequence; optional input and output
- Optional supplemental routines in the DLSODE package
- internal COMMON block

*Usage:

Communication between the user and the DLSODE package, for normal situations, is summarized here. This summary describes a subset of the available options. See "Long Description" for complete details, including optional communication, nonstandard options, and instructions for special situations.

A sample program is given in the "Examples" section.

Refer to the argument descriptions for the definitions of the quantities that appear in the following sample declarations.

```
For MF = 10,  
    PARAMETER (LRW = 20 + 16*NEQ,          LIW = 20)  
For MF = 21 or 22,
```

```

      PARAMETER (LRW = 22 + 9*NEQ + NEQ**2, LIW = 20 + NEQ)
For MF = 24 or 25,
      PARAMETER (LRW = 22 + 10*NEQ + (2*ML+MU)*NEQ,
*
*                                     LIW = 20 + NEQ)

      EXTERNAL F, JAC
      INTEGER NEQ, ITOL, ITASK, ISTATE, IOPT, LRW, IWORK(LIW),
*
*      LIW, MF
      DOUBLE PRECISION Y(NEQ), T, TOUT, RTOL, ATOL(NTOL), RWORK(LRW)

      CALL DLSODE (F, NEQ, Y, T, TOUT, ITOL, RTOL, ATOL, ITASK,
*
*      ISTATE, IOPT, RWORK, LRW, IWORK, LIW, JAC, MF)

```

***Arguments:**

F	:EXT	<p>Name of subroutine for right-hand-side vector f. This name must be declared EXTERNAL in calling program. The form of F must be:</p> <pre> SUBROUTINE F (NEQ, T, Y, YDOT) INTEGER NEQ DOUBLE PRECISION T, Y(NEQ), YDOT(NEQ) </pre> <p>The inputs are NEQ, T, Y. F is to set</p> $YDOT(i) = f(i, T, Y(1), Y(2), \dots, Y(NEQ)),$ <p style="text-align: right; margin-right: 100px;">$i = 1, \dots, NEQ.$</p>
NEQ	:IN	<p>Number of first-order ODE's.</p>
Y	:INOUT	<p>Array of values of the y(t) vector, of length NEQ.</p> <p>Input: For the first call, Y should contain the values of y(t) at t = T. (Y is an input variable only if ISTATE = 1.)</p> <p>Output: On return, Y will contain the values at the new t-value.</p>
T	:INOUT	<p>Value of the independent variable. On return it will be the current value of t (normally TOUT).</p>
TOUT	:IN	<p>Next point where output is desired (.NE. T).</p>
ITOL	:IN	<p>1 or 2 according as ATOL (below) is a scalar or an array.</p>
RTOL	:IN	<p>Relative tolerance parameter (scalar).</p>
ATOL	:IN	<p>Absolute tolerance parameter (scalar or array).</p> <p>If ITOL = 1, ATOL need not be dimensioned.</p> <p>If ITOL = 2, ATOL must be dimensioned at least NEQ.</p> <p>The estimated local error in Y(i) will be controlled so as to be roughly less (in magnitude) than</p> $EWT(i) = RTOL * ABS(Y(i)) + ATOL \quad \text{if ITOL} = 1, \text{ or}$ $EWT(i) = RTOL * ABS(Y(i)) + ATOL(i) \quad \text{if ITOL} = 2.$ <p>Thus the local error test passes if, in each component, either the absolute error is less than ATOL (or ATOL(i)), or the relative error is less than RTOL.</p>

Use $RTOL = 0.0$ for pure absolute error control, and use $ATOL = 0.0$ (or $ATOL(i) = 0.0$) for pure relative error control. Caution: Actual (global) errors may exceed these local tolerances, so choose them conservatively.

ITASK :IN Flag indicating the task DLSODE is to perform. Use ITASK = 1 for normal computation of output values of y at t = TOUT.

ISTATE:INOUT Index used for input and output to specify the state of the calculation.

Input:

- 1 This is the first call for a problem.
- 2 This is a subsequent call.

Output:

- 2 DLSODE was successful (otherwise, negative). Note that ISTATE need not be modified after a successful return.
- 1 Excess work done on this call (perhaps wrong MF).
- 2 Excess accuracy requested (tolerances too small).
- 3 Illegal input detected (see printed message).
- 4 Repeated error test failures (check all inputs).
- 5 Repeated convergence failures (perhaps bad Jacobian supplied or wrong choice of MF or tolerances).
- 6 Error weight became zero during problem (solution component i vanished, and ATOL or $ATOL(i) = 0.$).

IOPT :IN Flag indicating whether optional inputs are used:

- 0 No.
- 1 Yes. (See "Optional inputs" under "Long Description," Part 1.)

RWORK :WORK Real work array of length at least:

- 20 + 16*NEQ for MF = 10,
- 22 + 9*NEQ + NEQ**2 for MF = 21 or 22,
- 22 + 10*NEQ + (2*ML + MU)*NEQ for MF = 24 or 25.

LRW :IN Declared length of RWORK (in user's DIMENSION statement).

IWORK :WORK Integer work array of length at least:

- 20 for MF = 10,
- 20 + NEQ for MF = 21, 22, 24, or 25.

If MF = 24 or 25, input in IWORK(1),IWORK(2) the lower and upper Jacobian half-bandwidths ML,MU.

On return, IWORK contains information that may be of interest to the user:

Name	Location	Meaning
NST	IWORK(11)	Number of steps taken for the problem so

		far.
NFE	IWORK(12)	Number of f evaluations for the problem so far.
NJE	IWORK(13)	Number of Jacobian evaluations (and of matrix LU decompositions) for the problem so far.
NQU	IWORK(14)	Method order last used (successfully).
LENRW	IWORK(17)	Length of RWORK actually required. This is defined on normal returns and on an illegal input return for insufficient storage.
LENIW	IWORK(18)	Length of IWORK actually required. This is defined on normal returns and on an illegal input return for insufficient storage.
LIW	:IN	Declared length of IWORK (in user's DIMENSION statement).
JAC	:EXT	Name of subroutine for Jacobian matrix (MF = 21 or 24). If used, this name must be declared EXTERNAL in calling program. If not used, pass a dummy name. The form of JAC must be: <pre> SUBROUTINE JAC (NEQ, T, Y, ML, MU, PD, NROWPD) INTEGER NEQ, ML, MU, NROWPD DOUBLE PRECISION T, Y(NEQ), PD(NROWPD,NEQ) </pre> See item c, under "Description" below for more information about JAC.
MF	:IN	Method flag. Standard values are: 10 Nonstiff (Adams) method, no Jacobian used. 21 Stiff (BDF) method, user-supplied full Jacobian. 22 Stiff method, internally generated full Jacobian. 24 Stiff method, user-supplied banded Jacobian. 25 Stiff method, internally generated banded Jacobian.

***Description:**

DLSODE solves the initial value problem for stiff or nonstiff systems of first-order ODE's,

$$dy/dt = f(t,y) ,$$

or, in component form,

$$dy(i)/dt = f(i) = f(i,t,y(1),y(2),\dots,y(NEQ)) \quad (i = 1, \dots, NEQ) .$$

DLSODE is a package based on the GEAR and GEARB packages, and on the October 23, 1978, version of the tentative ODEPACK user interface standard, with minor modifications.

The steps in solving such a problem are as follows.

a. First write a subroutine of the form

```

SUBROUTINE F (NEQ, T, Y, YDOT)
      PMATH Reference Manual - 38

```

```

      INTEGER  NEQ
      DOUBLE PRECISION  T, Y(NEQ), YDOT(NEQ)

```

which supplies the vector function f by loading $YDOT(i)$ with $f(i)$.

- b. Next determine (or guess) whether or not the problem is stiff. Stiffness occurs when the Jacobian matrix df/dy has an eigenvalue whose real part is negative and large in magnitude compared to the reciprocal of the t span of interest. If the problem is nonstiff, use method flag $MF = 10$. If it is stiff, there are four standard choices for MF , and DLSODE requires the Jacobian matrix in some form. This matrix is regarded either as full ($MF = 21$ or 22), or banded ($MF = 24$ or 25). In the banded case, DLSODE requires two half-bandwidth parameters ML and MU . These are, respectively, the widths of the lower and upper parts of the band, excluding the main diagonal. Thus the band consists of the locations (i,j) with

$$i - ML \leq j \leq i + MU ,$$

and the full bandwidth is $ML + MU + 1$.

- c. If the problem is stiff, you are encouraged to supply the Jacobian directly ($MF = 21$ or 24), but if this is not feasible, DLSODE will compute it internally by difference quotients ($MF = 22$ or 25). If you are supplying the Jacobian, write a subroutine of the form

```

      SUBROUTINE  JAC (NEQ, T, Y, ML, MU, PD, NROWPD)
      INTEGER  NEQ, ML, MU, NRWOPD
      DOUBLE PRECISION  Y, Y(NEQ), PD(NROWPD,NEQ)

```

which provides df/dy by loading PD as follows:

- For a full Jacobian ($MF = 21$), load $PD(i,j)$ with $df(i)/dy(j)$, the partial derivative of $f(i)$ with respect to $y(j)$. (Ignore the ML and MU arguments in this case.)
 - For a banded Jacobian ($MF = 24$), load $PD(i-j+MU+1,j)$ with $df(i)/dy(j)$; i.e., load the diagonal lines of df/dy into the rows of PD from the top down.
 - In either case, only nonzero elements need be loaded.
- d. Write a main program that calls subroutine DLSODE once for each point at which answers are desired. This should also provide for possible use of logical unit 6 for output of error messages by DLSODE.

Before the first call to DLSODE, set $ISTATE = 1$, set Y and T to the initial values, and set $TOUT$ to the first output point. To continue the integration after a successful return, simply reset $TOUT$ and call DLSODE again. No other parameters need be reset.

*Examples:

The following is a simple example problem, with the coding needed for its solution by DLSODE. The problem is from chemical kinetics, and consists of the following three rate equations:

$$\begin{aligned}
 dy_1/dt &= -.04*y_1 + 1.E4*y_2*y_3 \\
 dy_2/dt &= .04*y_1 - 1.E4*y_2*y_3 - 3.E7*y_2**2
 \end{aligned}$$

$$dy_3/dt = 3.E7*y_2**2$$

on the interval from $t = 0.0$ to $t = 4.E10$, with initial conditions $y_1 = 1.0$, $y_2 = y_3 = 0$. The problem is stiff.

The following coding solves this problem with DLSODE, using $MF = 21$ and printing results at $t = .4, 4., \dots, 4.E10$. It uses $ITOL = 2$ and $ATOL$ much smaller for y_2 than for y_1 or y_3 because y_2 has much smaller values. At the end of the run, statistical quantities of interest are printed.

```

      EXTERNAL  FEX, JEX
      INTEGER  IOPT, IOUT, ISTATE, ITASK, ITOL, IWORK(23), LIW, LRW,
*            MF, NEQ
      DOUBLE PRECISION  ATOL(3), RTOL, RWORK(58), T, TOUT, Y(3)
      NEQ = 3
      Y(1) = 1.D0
      Y(2) = 0.D0
      Y(3) = 0.D0
      T = 0.D0
      TOUT = .4D0
      ITOL = 2
      RTOL = 1.D-4
      ATOL(1) = 1.D-6
      ATOL(2) = 1.D-10
      ATOL(3) = 1.D-6
      ITASK = 1
      ISTATE = 1
      IOPT = 0
      LRW = 58
      LIW = 23
      MF = 21
      DO 40 IOUT = 1,12
        CALL DLSODE (FEX, NEQ, Y, T, TOUT, ITOL, RTOL, ATOL, ITASK,
*                  ISTATE, IOPT, RWORK, LRW, IWORK, LIW, JEX, MF)
        WRITE(6,20)  T, Y(1), Y(2), Y(3)
20      FORMAT(' At t =',D12.4,'    y =',3D14.6)
        IF (ISTATE .LT. 0)  GO TO 80
40      TOUT = TOUT*10.D0
        WRITE(6,60)  IWORK(11), IWORK(12), IWORK(13)
60      FORMAT('/' No. steps =',i4,',  No. f-s =',i4,',  No. J-s =',i4)
        STOP
80      WRITE(6,90)  ISTATE
90      FORMAT('Error halt.. ISTATE =',I3)
        STOP
      END

      SUBROUTINE  FEX (NEQ, T, Y, YDOT)
      INTEGER  NEQ
      DOUBLE PRECISION  T, Y(3), YDOT(3)
      YDOT(1) = -.04D0*Y(1) + 1.D4*Y(2)*Y(3)
      YDOT(3) = 3.D7*Y(2)*Y(2)
      YDOT(2) = -YDOT(1) - YDOT(3)
      RETURN
      END

      SUBROUTINE  JEX (NEQ, T, Y, ML, MU, PD, NRPD)
      INTEGER  NEQ, ML, MU, NRPD
      DOUBLE PRECISION  T, Y(3), PD(NRPD,3)
      PD(1,1) = -.04D0

```



```

PD(1,2) = 1.D4*Y(3)
PD(1,3) = 1.D4*Y(2)
PD(2,1) = .04D0
PD(2,3) = -PD(1,3)
PD(3,2) = 6.D7*Y(2)
PD(2,2) = -PD(1,2) - PD(3,2)
RETURN
END

```

The output from this program (on a Cray-1 in single precision) is as follows.

At t =	4.0000e-01	y =	9.851726e-01	3.386406e-05	1.479357e-02
At t =	4.0000e+00	y =	9.055142e-01	2.240418e-05	9.446344e-02
At t =	4.0000e+01	y =	7.158050e-01	9.184616e-06	2.841858e-01
At t =	4.0000e+02	y =	4.504846e-01	3.222434e-06	5.495122e-01
At t =	4.0000e+03	y =	1.831701e-01	8.940379e-07	8.168290e-01
At t =	4.0000e+04	y =	3.897016e-02	1.621193e-07	9.610297e-01
At t =	4.0000e+05	y =	4.935213e-03	1.983756e-08	9.950648e-01
At t =	4.0000e+06	y =	5.159269e-04	2.064759e-09	9.994841e-01
At t =	4.0000e+07	y =	5.306413e-05	2.122677e-10	9.999469e-01
At t =	4.0000e+08	y =	5.494530e-06	2.197825e-11	9.999945e-01
At t =	4.0000e+09	y =	5.129458e-07	2.051784e-12	9.999995e-01
At t =	4.0000e+10	y =	-7.170603e-08	-2.868241e-13	1.000000e+00

No. steps = 330, No. f-s = 405, No. J-s = 69

***Accuracy:**

The accuracy of the solution depends on the choice of tolerances RTOL and ATOL. Actual (global) errors may exceed these local tolerances, so choose them conservatively.

***Cautions:**

The work arrays should not be altered between calls to DLSODE for the same problem, except possibly for the conditional and optional inputs.

***Portability:**

Since NEQ is dimensioned inside DLSODE, some compilers may object to a call to DLSODE with NEQ a scalar variable. In this event, use DIMENSION NEQ(1). Similar remarks apply to RTOL and ATOL.

Note to Cray users:

For maximum efficiency, use the CFT77 compiler. Appropriate compiler optimization directives have been inserted for CFT77 (but not CIVIC).

NOTICE: If moving the DLSODE source code to other systems, contact the author for notes on nonstandard Fortran usage, COMMON block, and other installation details.

***Reference:**

Alan C. Hindmarsh, "ODEPACK, a systematized collection of ODE solvers," in Scientific Computing, R. S. Stepleman, et al., Eds. (North-Holland, Amsterdam, 1983), pp. 55-64.

***Long Description:**

The following complete description of the user interface to DLSODE consists of four parts:

1. The call sequence to subroutine DLSODE, which is a driver routine for the solver. This includes descriptions of both the call sequence arguments and user-supplied routines. Following these descriptions is a description of optional inputs available through the call sequence, and then a description of optional outputs in the work arrays.
2. Descriptions of other routines in the DLSODE package that may be (optionally) called by the user. These provide the ability to alter error message handling, save and restore the internal COMMON, and obtain specified derivatives of the solution $y(t)$.
3. Descriptions of COMMON block to be declared in overlay or similar environments, or to be saved when doing an interrupt of the problem and continued solution later.
4. Description of two routines in the DLSODE package, either of which the user may replace with his own version, if desired. These relate to the measurement of errors.

Part 1. Call Sequence

Arguments

The call sequence parameters used for input only are

F, NEQ, TOUT, ITOL, RTOL, ATOL, ITASK, IOPT, LRW, LIW, JAC, MF,

and those used for both input and output are

Y, T, ISTATE.

The work arrays RWORK and IWORK are also used for conditional and optional inputs and optional outputs. (The term output here refers to the return from subroutine DLSODE to the user's calling program.)

The legality of input parameters will be thoroughly checked on the initial call for the problem, but not checked thereafter unless a change in input parameters is flagged by ISTATE = 3 on input.

The descriptions of the call arguments are as follows.

F The name of the user-supplied subroutine defining the ODE system. The system must be put in the first-order form $dy/dt = f(t,y)$, where f is a vector-valued function of the scalar t and the vector y . Subroutine F is to compute the function f . It is to have the form

```
SUBROUTINE F (NEQ, T, Y, YDOT)
  DOUBLE PRECISION  Y(NEQ), YDOT(NEQ)
```

where NEQ, T, and Y are input, and the array YDOT = $f(T,Y)$ is output. Y and YDOT are arrays of length NEQ. Subroutine F should not alter $Y(1), \dots, Y(NEQ)$. F must be declared EXTERNAL in the calling program.

Subroutine F may access user-defined quantities in

NEQ(2),... and/or in Y(NEQ(1)+1),..., if NEQ is an array (dimensioned in F) and/or Y has length exceeding NEQ(1). See the descriptions of NEQ and Y below.

If quantities computed in the F routine are needed externally to DLSODE, an extra call to F should be made for this purpose, for consistent and accurate results. If only the derivative dy/dt is needed, use DINTDY instead.

NEQ The size of the ODE system (number of first-order ordinary differential equations). Used only for input. NEQ may be decreased, but not increased, during the problem. If NEQ is decreased (with ISTATE = 3 on input), the remaining components of Y should be left undisturbed, if these are to be accessed in F and/or JAC.

Normally, NEQ is a scalar, and it is generally referred to as a scalar in this user interface description. However, NEQ may be an array, with NEQ(1) set to the system size. (The DLSODE package accesses only NEQ(1).) In either case, this parameter is passed as the NEQ argument in all calls to F and JAC. Hence, if it is an array, locations NEQ(2),... may be used to store other integer data and pass it to F and/or JAC. Subroutines F and/or JAC must include NEQ in a DIMENSION statement in that case.

Y A real array for the vector of dependent variables, of length NEQ or more. Used for both input and output on the first call (ISTATE = 1), and only for output on other calls. On the first call, Y must contain the vector of initial values. On output, Y contains the computed solution vector, evaluated at T. If desired, the Y array may be used for other purposes between calls to the solver.

This array is passed as the Y argument in all calls to F and JAC. Hence its length may exceed NEQ, and locations Y(NEQ+1),... may be used to store other real data and pass it to F and/or JAC. (The DLSODE package accesses only Y(1),...,Y(NEQ).)

T The independent variable. On input, T is used only on the first call, as the initial point of the integration. On output, after each call, T is the value at which a computed solution Y is evaluated (usually the same as TOUT). On an error return, T is the farthest point reached.

TOUT The next value of T at which a computed solution is desired. Used only for input.

When starting the problem (ISTATE = 1), TOUT may be equal to T for one call, then should not equal T for the next call. For the initial T, an input value of TOUT .NE. T is used in order to determine the direction of the integration (i.e., the algebraic sign of the step sizes) and the rough scale of the problem. Integration in either direction (forward or backward in T) is permitted.

If ITASK = 2 or 5 (one-step modes), TOUT is ignored after the first call (i.e., the first call with TOUT .NE. T). Otherwise, TOUT is required on every call.

If ITASK = 1, 3, or 4, the values of TOUT need not be monotone, but a value of TOUT which backs up is limited to the current internal T interval, whose endpoints are TCUR - HU and TCUR. (See "Optional Outputs" below for TCUR and HU.)

ITOL An indicator for the type of error control. See description below under ATOL. Used only for input.

RTOL A relative error tolerance parameter, either a scalar or an array of length NEQ. See description below under ATOL. Input only.

ATOL An absolute error tolerance parameter, either a scalar or an array of length NEQ. Input only.

The input parameters ITOL, RTOL, and ATOL determine the error control performed by the solver. The solver will control the vector $e = (e(i))$ of estimated local errors in Y, according to an inequality of the form

$$\text{rms-norm of } (e(i)/EWT(i)) \leq 1,$$

where

$$EWT(i) = RTOL(i)*ABS(Y(i)) + ATOL(i),$$

and the rms-norm (root-mean-square norm) here is

$$\text{rms-norm}(v) = \text{SQRT}(\text{sum } v(i)**2 / \text{NEQ}).$$

Here EWT = (EWT(i)) is a vector of weights which must always be positive, and the values of RTOL and ATOL should all be nonnegative. The following table gives the types (scalar/array) of RTOL and ATOL, and the corresponding form of EWT(i).

ITOL	RTOL	ATOL	EWT(i)
----	-----	-----	-----
1	scalar	scalar	$RTOL*ABS(Y(i)) + ATOL$
2	scalar	array	$RTOL*ABS(Y(i)) + ATOL(i)$
3	array	scalar	$RTOL(i)*ABS(Y(i)) + ATOL$
4	array	array	$RTOL(i)*ABS(Y(i)) + ATOL(i)$

When either of these parameters is a scalar, it need not be dimensioned in the user's calling program.

If none of the above choices (with ITOL, RTOL, and ATOL fixed throughout the problem) is suitable, more general error controls can be obtained by substituting user-supplied routines for the setting of EWT and/or for the norm calculation. See Part 4 below.

If global errors are to be estimated by making a repeated

run on the same problem with smaller tolerances, then all components of RTOL and ATOL (i.e., of EWT) should be scaled down uniformly.

ITASK An index specifying the task to be performed. Input only. ITASK has the following values and meanings:

- 1 Normal computation of output values of $y(t)$ at $t = TOUT$ (by overshooting and interpolating).
- 2 Take one step only and return.
- 3 Stop at the first internal mesh point at or beyond $t = TOUT$ and return.
- 4 Normal computation of output values of $y(t)$ at $t = TOUT$ but without overshooting $t = TCRIT$. TCRIT must be input as RWORK(1). TCRIT may be equal to or beyond TOUT, but not behind it in the direction of integration. This option is useful if the problem has a singularity at or beyond $t = TCRIT$.
- 5 Take one step, without passing TCRIT, and return. TCRIT must be input as RWORK(1).

Note: If ITASK = 4 or 5 and the solver reaches TCRIT (within roundoff), it will return $T = TCRIT$ (exactly) to indicate this (unless ITASK = 4 and TOUT comes before TCRIT, in which case answers at $T = TOUT$ are returned first).

ISTATE An index used for input and output to specify the state of the calculation.

On input, the values of ISTATE are as follows:

- 1 This is the first call for the problem (initializations will be done). See "Note" below.
- 2 This is not the first call, and the calculation is to continue normally, with no change in any input parameters except possibly TOUT and ITASK. (If ITOL, RTOL, and/or ATOL are changed between calls with ISTATE = 2, the new values will be used but not tested for legality.)
- 3 This is not the first call, and the calculation is to continue normally, but with a change in input parameters other than TOUT and ITASK. Changes are allowed in NEQ, ITOL, RTOL, ATOL, IOPT, LRW, LIW, MF, ML, MU, and any of the optional inputs except H0. (See IWORK description for ML and MU.)

Note: A preliminary call with $TOUT = T$ is not counted as a first call here, as no initialization or checking of input is done. (Such a call is sometimes useful for the purpose of outputting the initial conditions.) Thus the first call for which $TOUT \neq T$ requires ISTATE = 1 on input.

On output, ISTATE has the following values and meanings:

- 1 Nothing was done, as TOUT was equal to T with ISTATE = 1 on input.
- 2 The integration was performed successfully.
- 1 An excessive amount of work (more than MXSTEP steps) was done on this call, before completing the requested task, but the integration was otherwise successful as far as T. (MXSTEP is an optional input

and is normally 500.) To continue, the user may simply reset ISTATE to a value >1 and call again (the excess work step counter will be reset to 0). In addition, the user may increase MXSTEP to avoid this error return; see "Optional Inputs" below.

- 2 Too much accuracy was requested for the precision of the machine being used. This was detected before completing the requested task, but the integration was successful as far as T. To continue, the tolerance parameters must be reset, and ISTATE must be set to 3. The optional output TOLSF may be used for this purpose. (Note: If this condition is detected before taking any steps, then an illegal input return (ISTATE = -3) occurs instead.)
- 3 Illegal input was detected, before taking any integration steps. See written message for details. (Note: If the solver detects an infinite loop of calls to the solver with illegal input, it will cause the run to stop.)
- 4 There were repeated error-test failures on one attempted step, before completing the requested task, but the integration was successful as far as T. The problem may have a singularity, or the input may be inappropriate.
- 5 There were repeated convergence-test failures on one attempted step, before completing the requested task, but the integration was successful as far as T. This may be caused by an inaccurate Jacobian matrix, if one is being used.
- 6 EWT(i) became zero for some i during the integration. Pure relative error control (ATOL(i)=0.0) was requested on a variable which has now vanished. The integration was successful as far as T.

Note: Since the normal output value of ISTATE is 2, it does not need to be reset for normal continuation. Also, since a negative input value of ISTATE will be regarded as illegal, a negative output value requires the user to change it, and possibly other inputs, before calling the solver again.

IOPT An integer flag to specify whether any optional inputs are being used on this call. Input only. The optional inputs are listed under a separate heading below.

0 No optional inputs are being used. Default values will be used in all cases.

1 One or more optional inputs are being used.

RWORK A real working array (double precision). The length of RWORK must be at least

$$20 + NYH*(MAXORD + 1) + 3*NEQ + LWM$$

where

NYH = the initial value of NEQ,
 MAXORD = 12 (if METH = 1) or 5 (if METH = 2) (unless a smaller value is given as an optional input),
 LWM = 0 if MITER = 0,
 LWM = NEQ**2 + 2 if MITER = 1 or 2,
 LWM = NEQ + 2 if MITER = 3, and

$$\text{LWM} = (2 * \text{ML} + \text{MU} + 1) * \text{NEQ} + 2$$

if MITER = 4 or 5.

(See the MF description below for METH and MITER.)

Thus if MAXORD has its default value and NEQ is constant, this length is:

20 + 16*NEQ	for MF = 10,
22 + 16*NEQ + NEQ**2	for MF = 11 or 12,
22 + 17*NEQ	for MF = 13,
22 + 17*NEQ + (2*ML + MU)*NEQ	for MF = 14 or 15,
20 + 9*NEQ	for MF = 20,
22 + 9*NEQ + NEQ**2	for MF = 21 or 22,
22 + 10*NEQ	for MF = 23,
22 + 10*NEQ + (2*ML + MU)*NEQ	for MF = 24 or 25.

The first 20 words of RWORK are reserved for conditional and optional inputs and optional outputs.

The following word in RWORK is a conditional input:
 RWORK(1) = TCRIT, the critical value of t which the solver is not to overshoot. Required if ITASK is 4 or 5, and ignored otherwise. See ITASK.

LRW The length of the array RWORK, as declared by the user.
 (This will be checked by the solver.)

IWORK An integer work array. Its length must be at least
 20 if MITER = 0 or 3 (MF = 10, 13, 20, 23), or
 20 + NEQ otherwise (MF = 11, 12, 14, 15, 21, 22, 24, 25).
 (See the MF description below for MITER.) The first few
 words of IWORK are used for conditional and optional
 inputs and optional outputs.

The following two words in IWORK are conditional inputs:
 IWORK(1) = ML These are the lower and upper half-
 IWORK(2) = MU bandwidths, respectively, of the banded
 Jacobian, excluding the main diagonal.
 The band is defined by the matrix locations
 (i,j) with $i - \text{ML} \leq j \leq i + \text{MU}$. ML and MU
 must satisfy $0 \leq \text{ML}, \text{MU} \leq \text{NEQ} - 1$. These are
 required if MITER is 4 or 5, and ignored
 otherwise. ML and MU may in fact be the band
 parameters for a matrix to which df/dy is only
 approximately equal.

LIW The length of the array IWORK, as declared by the user.
 (This will be checked by the solver.)

Note: The work arrays must not be altered between calls to DLSODE for the same problem, except possibly for the conditional and optional inputs, and except for the last 3*NEQ words of RWORK. The latter space is used for internal scratch space, and so is available for use by the user outside DLSODE between calls, if desired (but not for use by F or JAC).

JAC The name of the user-supplied routine (MITER = 1 or 4) to compute the Jacobian matrix, df/dy , as a function of the scalar t and the vector y. (See the MF description below for MITER.) It is to have the form

```

SUBROUTINE JAC (NEQ, T, Y, ML, MU, PD, NROWPD)
DOUBLE PRECISION Y(NEQ), PD(NROWPD,NEQ)

```

where NEQ, T, Y, ML, MU, and NROWPD are input and the array PD is to be loaded with partial derivatives (elements of the Jacobian matrix) on output. PD must be given a first dimension of NROWPD. T and Y have the same meaning as in subroutine F.

In the full matrix case (MITER = 1), ML and MU are ignored, and the Jacobian is to be loaded into PD in columnwise manner, with $df(i)/dy(j)$ loaded into PD(i,j).

In the band matrix case (MITER = 4), the elements within the band are to be loaded into PD in columnwise manner, with diagonal lines of df/dy loaded into the rows of PD. Thus $df(i)/dy(j)$ is to be loaded into PD(i-j+MU+1,j). ML and MU are the half-bandwidth parameters (see IWORK). The locations in PD in the two triangular areas which correspond to nonexistent matrix elements can be ignored or loaded arbitrarily, as they are overwritten by DLSODE.

JAC need not provide df/dy exactly. A crude approximation (possibly with a smaller bandwidth) will do.

In either case, PD is preset to zero by the solver, so that only the nonzero elements need be loaded by JAC. Each call to JAC is preceded by a call to F with the same arguments NEQ, T, and Y. Thus to gain some efficiency, intermediate quantities shared by both calculations may be saved in a user COMMON block by F and not recomputed by JAC, if desired. Also, JAC may alter the Y array, if desired. JAC must be declared EXTERNAL in the calling program.

Subroutine JAC may access user-defined quantities in NEQ(2),... and/or in Y(NEQ(1)+1),... if NEQ is an array (dimensioned in JAC) and/or Y has length exceeding NEQ(1). See the descriptions of NEQ and Y above.

MF The method flag. Used only for input. The legal values of MF are 10, 11, 12, 13, 14, 15, 20, 21, 22, 23, 24, and 25. MF has decimal digits METH and MITER:

MF = 10*METH + MITER .

METH indicates the basic linear multistep method:

- 1 Implicit Adams method.
- 2 Method based on backward differentiation formulas (BDF's).

MITER indicates the corrector iteration method:

- 0 Functional iteration (no Jacobian matrix is involved).
- 1 Chord iteration with a user-supplied full (NEQ by NEQ) Jacobian.
- 2 Chord iteration with an internally generated (difference quotient) full Jacobian (using NEQ extra calls to F per df/dy value).
- 3 Chord iteration with an internally generated diagonal Jacobian approximation (using one extra call

- to F per df/dy evaluation).
- 4 Chord iteration with a user-supplied banded Jacobian.
 - 5 Chord iteration with an internally generated banded Jacobian (using ML + MU + 1 extra calls to F per df/dy evaluation).

If MITER = 1 or 4, the user must supply a subroutine JAC (the name is arbitrary) as described above under JAC. For other values of MITER, a dummy argument can be used.

Optional Inputs

The following is a list of the optional inputs provided for in the call sequence. (See also Part 2.) For each such input variable, this table lists its name as used in this documentation, its location in the call sequence, its meaning, and the default value. The use of any of these inputs requires IOPT = 1, and in that case all of these inputs are examined. A value of zero for any of these optional inputs will cause the default value to be used. Thus to use a subset of the optional inputs, simply preload locations 5 to 10 in RWORK and IWORK to 0.0 and 0 respectively, and then set those of interest to nonzero values.

Name	Location	Meaning and default value
-----	-----	-----
H0	RWORK(5)	Step size to be attempted on the first step. The default value is determined by the solver.
HMAX	RWORK(6)	Maximum absolute step size allowed. The default value is infinite.
HMIN	RWORK(7)	Minimum absolute step size allowed. The default value is 0. (This lower bound is not enforced on the final step before reaching TCRIT when ITASK = 4 or 5.)
MAXORD	IWORK(5)	Maximum order to be allowed. The default value is 12 if METH = 1, and 5 if METH = 2. (See the MF description above for METH.) If MAXORD exceeds the default value, it will be reduced to the default value. If MAXORD is changed during the problem, it may cause the current order to be reduced.
MXSTEP	IWORK(6)	Maximum number of (internally defined) steps allowed during one call to the solver. The default value is 500.
MXHNIL	IWORK(7)	Maximum number of messages printed (per problem) warning that $T + H = T$ on a step (H = step size). This must be positive to result in a nondefault value. The default value is 10.

Optional Outputs

As optional additional output from DLSODE, the variables listed below are quantities related to the performance of DLSODE which are available to the user. These are communicated by way of the work arrays, but also have internal mnemonic names as shown. Except where stated otherwise, all of these outputs are defined on any successful return from DLSODE, and on any return with ISTATE = -1, -2, -4, -5, or -6. On an illegal input return (ISTATE = -3), they will be unchanged from their existing values (if any), except possibly for TOLSF, LENRW, and LENIW. On any error return,

outputs relevant to the error will be defined, as noted below.

Name	Location	Meaning
-----	-----	-----
HU	RWORK(11)	Step size in t last used (successfully).
HCUR	RWORK(12)	Step size to be attempted on the next step.
TCUR	RWORK(13)	Current value of the independent variable which the solver has actually reached, i.e., the current internal mesh point in t. On output, TCUR will always be at least as far as the argument T, but may be farther (if interpolation was done).
TOLSF	RWORK(14)	Tolerance scale factor, greater than 1.0, computed when a request for too much accuracy was detected (ISTATE = -3 if detected at the start of the problem, ISTATE = -2 otherwise). If ITOL is left unaltered but RTOL and ATOL are uniformly scaled up by a factor of TOLSF for the next call, then the solver is deemed likely to succeed. (The user may also ignore TOLSF and alter the tolerance parameters in any other way appropriate.)
NST	IWORK(11)	Number of steps taken for the problem so far.
NFE	IWORK(12)	Number of F evaluations for the problem so far.
NJE	IWORK(13)	Number of Jacobian evaluations (and of matrix LU decompositions) for the problem so far.
NQU	IWORK(14)	Method order last used (successfully).
NQCUR	IWORK(15)	Order to be attempted on the next step.
IMXER	IWORK(16)	Index of the component of largest magnitude in the weighted local error vector (e(i)/EWT(i)), on an error return with ISTATE = -4 or -5.
LENRW	IWORK(17)	Length of RWORK actually required. This is defined on normal returns and on an illegal input return for insufficient storage.
LENIW	IWORK(18)	Length of IWORK actually required. This is defined on normal returns and on an illegal input return for insufficient storage.

The following two arrays are segments of the RWORK array which may also be of interest to the user as optional outputs. For each array, the table below gives its internal name, its base address in RWORK, and its description.

Name	Base address	Description
-----	-----	-----
YH	21	The Nordsieck history array, of size NYH by (NQCUR + 1), where NYH is the initial value of NEQ. For j = 0,1,...,NQCUR, column j + 1 of YH contains HCUR**j/factorial(j) times the jth derivative of the interpolating polynomial currently representing the solution, evaluated at t = TCUR.
ACOR	LENRW-NEQ+1	Array of size NEQ used for the accumulated corrections on each step, scaled on output to represent the estimated local error in Y on the last step. This is the vector e in the description of the error control. It is defined only on successful return from DLSODE.

Part 2. Other Callable Routines

The following are optional calls which the user may make to gain additional capabilities in conjunction with DLSODE.

Form of call	Function
-----	-----
CALL XSETUN(LUN)	Set the logical unit number, LUN, for output of messages from DLSODE, if the default is not desired. The default value of LUN is 6. This call may be made at any time and will take effect immediately.
CALL XSETF(MFLAG)	Set a flag to control the printing of messages by DLSODE. MFLAG = 0 means do not print. (Danger: this risks losing valuable information.) MFLAG = 1 means print (the default). This call may be made at any time and will take effect immediately.
CALL DSRCOM(RSAV,ISAV,JOB)	Saves and restores the contents of the internal COMMON blocks used by DLSODE (see Part 3 below). RSAV must be a real array of length 218 or more, and ISAV must be an integer array of length 37 or more. JOB = 1 means save COMMON into RSAV/ISAV. JOB = 2 means restore COMMON from same. DSRCOM is useful if one is interrupting a run and restarting later, or alternating between two or more problems solved with DLSODE.
CALL DINTDY(,,,,,) (see below)	Provide derivatives of y, of various orders, at a specified point t, if desired. It may be called only after a successful return from DLSODE. Detailed instructions follow.

Detailed instructions for using DINTDY

The form of the CALL is:

CALL DINTDY (T, K, RWORK(21), NYH, DKY, IFLAG)

The input parameters are:

T	Value of independent variable where answers are desired (normally the same as the T last returned by DLSODE). For valid results, T must lie between TCUR - HU and TCUR. (See "Optional Outputs" above for TCUR and HU.)
K	Integer order of the derivative desired. K must satisfy $0 \leq K \leq \text{NQCUR}$, where NQCUR is the current order (see "Optional Outputs"). The capability corresponding to $K = 0$, i.e., computing $y(t)$, is already provided by DLSODE directly. Since $\text{NQCUR} \geq 1$, the first derivative dy/dt is always available with DINTDY.
RWORK(21)	The base address of the history array YH.
NYH	Column length of YH, equal to the initial value of NEQ.

The output parameters are:

DKY Real array of length NEQ containing the computed value
 of the Kth derivative of $y(t)$.
IFLAG Integer flag, returned as 0 if K and T were legal,
 -1 if K was illegal, and -2 if T was illegal.
 On an error return, a message is also written.

Part 3. Common Blocks

If DLSODE is to be used in an overlay situation, the user must declare, in the primary overlay, the variables in:

- (1) the call sequence to DLSODE,
- (2) the internal COMMON block /DLS001/, of length 255
(218 double precision words followed by 37 integer words).

If DLSODE is used on a system in which the contents of internal COMMON blocks are not preserved between calls, the user should declare the above COMMON block in his main program to insure that its contents are preserved.

If the solution of a given problem by DLSODE is to be interrupted and then later continued, as when restarting an interrupted run or alternating between two or more problems, the user should save, following the return from the last DLSODE call prior to the interruption, the contents of the call sequence variables and the internal COMMON block, and later restore these values before the next DLSODE call for that problem. In addition, if XSETUN and/or XSETF was called for non-default handling of error messages, then these calls must be repeated. To save and restore the COMMON block, use subroutine DSRCOM (see Part 2 above).

Part 4. Optionally Replaceable Solver Routines

Below are descriptions of two routines in the DLSODE package which relate to the measurement of errors. Either routine can be replaced by a user-supplied version, if desired. However, since such a replacement may have a major impact on performance, it should be done only when absolutely necessary, and only with great caution. (Note: The means by which the package version of a routine is superseded by the user's version may be system-dependent.)

DEWSET

The following subroutine is called just before each internal integration step, and sets the array of error weights, EWT, as described under ITOL/RTOL/ATOL above:

```
SUBROUTINE DEWSET (NEQ, ITOL, RTOL, ATOL, YCUR, EWT)
```

where NEQ, ITOL, RTOL, and ATOL are as in the DLSODE call sequence, YCUR contains the current dependent variable vector, and EWT is the array of weights set by DEWSET.

If the user supplies this subroutine, it must return in EWT(i) (i = 1,...,NEQ) a positive quantity suitable for comparing errors in Y(i) to. The EWT array returned by DEWSET is passed to the DVNORM routine (see below), and also used by DLSODE in the computation of the optional output IMXER, the diagonal Jacobian approximation, and the increments for difference quotient Jacobians.

In the user-supplied version of DEWSET, it may be desirable to use the current values of derivatives of y. Derivatives up to order NQ are available from the history array YH, described above under optional outputs. In DEWSET, YH is identical to the YCUR array, extended to NQ + 1 columns with a column length of NYH and scale factors of $H^{*j}/\text{factorial}(j)$. On the first call for the problem, given by NST = 0, NQ is 1 and H is temporarily set to 1.0. The quantities NQ, NYH, H, and NST can be obtained by including in DEWSET the statements:

```
DOUBLE PRECISION  RLS
COMMON /DLS001/ RLS(218),ILS(37)
NQ = ILS(33)
NYH = ILS(12)
NST = ILS(34)
H = RLS(212)
```

Thus, for example, the current value of dy/dt can be obtained as YCUR(NYH+i)/H (i=1,...,NEQ) (and the division by H is unnecessary when NST = 0).

DVNORM

DVNORM is a real function routine which computes the weighted root-mean-square norm of a vector v:

```
d = DVNORM (n, v, w)
```

where:

```
n = the length of the vector,
v = real array of length n containing the vector,
w = real array of length n containing weights,
d = SQRT( (1/n) * sum(v(i)*w(i))**2 ).
```

DVNORM is called with n = NEQ and with w(i) = 1.0/EWT(i), where EWT is as set by subroutine DEWSET.

If the user supplies this function, it should return a nonnegative value of DVNORM suitable for use in the error control in DLSODE. None of the arguments should be altered by DVNORM. For example, a user-supplied DVNORM routine might:

- Substitute a max-norm of (v(i)*w(i)) for the rms-norm, or
- Ignore some components of v in the norm, with the effect of suppressing the error control on those components of Y.

```
-----
***REFERENCES  Alan C. Hindmarsh, "ODEPACK, a systematized collection
                of ODE solvers", in Scientific Computing, R. S.
                Stepleman, et al. (Eds.), (North-Holland, Amsterdam,
                1983), pp. 55-64.
***ROUTINES CALLED  DEWSET, DINTDY, DUMACH, DSTODE, DVNORM, XERRWD
***COMMON BLOCKS    DLS001
***REVISION HISTORY  (YYMMDD)
```

```
791129  DATE WRITTEN
***END PROLOGUE  DLSODE
```

DMAXAF

```
DOUBLE PRECISION FUNCTION DMAXAF (ARRAY, IFIRST, ILAST, ISTRID,
+                                IMAX)
***BEGIN PROLOGUE  DMAXAF
***PURPOSE  Maximum value in a one-dimensional array.
***LIBRARY   PMATH
***CATEGORY  N5A
***TYPE      DOUBLE PRECISION (SMAXAF-S, DMAXAF-D, AMAXF8-8, IMAXAF-I)
***KEYWORDS  MAXIMUM
***AUTHOR   Painter, Jeffrey F., (LLNL)
            Revised by:
            Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine AMAXAF.)
*Usage:
    INTEGER  IFIRST, ILAST, ISTRID, IMAX
    DOUBLE PRECISION  ARRAY(n), AMAX, DMAXAF
    AMAX = DMAXAF (ARRAY, IFIRST, ILAST, ISTRID, IMAX)

*Arguments:
    ARRAY:IN   Real array to be searched.
               n, the dimension of the array, must be no less than
               ILAST.

    IFIRST:IN  First subscript in the array to be searched.

    ILAST :IN  Last subscript in the array to be searched.

    ISTRID:IN  Increment (stride) between successive locations that
               are to be searched.

    IMAX :OUT  Index of the maximum value in the array, i.e., the
               ordinal position of the value in the array.

*Function Return Values:
    AMAX :      Maximum value in the array.

*Description:
    DMAXAF finds the maximum value in a one-dimensional real array,
    and returns its index.  In case of multiple maxima, the last
    index found is returned.

*Cautions:
    The array is assumed to be subscripted from 1.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
    830401  DATE WRITTEN (J. F. Painter)
    930930  Converted old UNICOS names to S- or I-names.  (DBP)
    931005  Augmented list of equivalent routines.  (FNF)
    931018  Produced double precision version.  (DBP)
    940421  Corrected category.  (FNF)
    940727  Added preprocessor directives for REAL*8 entries.  (FNF)
    951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  DMAXAF
```

DMEANF

```
DOUBLE PRECISION FUNCTION DMEANF (A, N)
***BEGIN PROLOGUE  DMEANF
***PURPOSE  Mean of a one-dimensional real array.
***LIBRARY   PMATH
***CATEGORY  L1A
***TYPE      DOUBLE PRECISION (SMEANF-S, DMEANF-D, AMEAN8-8)
***KEYWORDS  ELEMENTARY STATISTICS, MEAN
***AUTHOR   Unknown, Name (LLNL/USD/NMG)
             Durst, Mark J. (LLNL/CMRD/SPG)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine AMEANF.)
*Usage:
    INTEGER  N
    DOUBLE PRECISION  ANS, A(N)
    ANS = DMEANF (A, N)

*Arguments:
    A :IN  Array of input values.
    N :IN  Number of elements in A.

*Function Return Values:
    ANS      The mean of the values in A.

*Description:
    DMEANF calculates the mean of the N values contained in A.

*See Also:
    For a vector of means, see DMEANV.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
    830812  DATE WRITTEN
            (The above is the date of LCSD-442,Rev.1 and is undoubtedly
            a significant underestimate of the age of this routine.)
    890223  Added SLATEC/LDOC prologue. (FNF)
    890518  Modified sequence numbers to fit in columns 73-80. (FNF)
    920319  Updated with prologue edited 891025 by G. Shaw for manual.
    930930  Converted old UNICOS names to S- or I-names. (DBP)
    931004  Corrected name conversion errors. (FNF)
    931005  Corrected list of equivalent routines and made sure that all
            variables are declared. (FNF)
    931018  Produced double precision version. (DBP)
    931026  Minor change to reduce single/double differences. (FNF)
    940727  Added preprocessor directives for REAL*8 entries. (FNF)
    951010  Corrected LIBRARY line. (FNF)
***END PROLOGUE  DMEANF
```


DMEANV

```
SUBROUTINE DMEANV (A, N, M, AV)
***BEGIN PROLOGUE  DMEANV
***PURPOSE  Mean vector of a two-dimensional real array.
             Calculates the means of N observations on each of M
             variables.
***LIBRARY    PMATH
***CATEGORY   L1B
***TYPE       DOUBLE PRECISION (SMEANV-S, DMEANV-D, MEANV8-8)
***KEYWORDS   ELEMENTARY STATISTICS, MEAN, VECTOR
***AUTHOR    Unknown, Name (LLNL/USD/NMG)
             Durst, Mark J. (LLNL/CMRD/SPG)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
             (Portable version of Cray MATHLIB routine AMEANV.)
*Usage:
      INTEGER  N, M
      DOUBLE PRECISION  A(N,M), AV(M)
      CALL DMEANV (A, N, M, AV)

*Arguments:
      A :IN      N by M array of N observations on M variables.
      N :IN      Row dimension of A.
      M :IN      Column dimension of A.
      AV:OUT     Array containing the values of the means, i.e.,

                   N
      AV(j)  =  sum  A(i,j) / N ,    j = 1,...,M.
                   i=1

*Description:
      DMEANV calculates the means of the N observations on each of M
      variables contained in the columns of A.

      The result AV is mathematically equivalent to applying DMEANF to
      each of the columns of A, but DMEANV should be faster.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
             (The above is the date of LCSD-442,Rev.1 and is undoubtedly
             a significant underestimate of the age of this routine.)
      890223  Added SLATEC/LDOC prologue. (FNF)
      890518  Modified sequence numbers to fit in columns 73-80. (FNF)
      920319  Updated with prologue edited 891025 by G. Shaw for manual.
***END PROLOGUE  DMEANV
```

DMEDF

```
DOUBLE PRECISION FUNCTION DMEDF (A, N, WK)
***BEGIN PROLOGUE  DMEDF
***PURPOSE  Median of a one-dimensional real array.
***LIBRARY   PMATH
***CATEGORY  L1A
***TYPE      DOUBLE PRECISION (SMEDF-S, DMEDF-D, AMED8-8)
***KEYWORDS  ELEMENTARY STATISTICS, MEDIAN
***AUTHOR    Unknown, Name (LLNL/USD/NMG)
              Durst, Mark J. (LLNL/CMRD/SPG)
              Currently responsible:
              Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine AMEDF.)
*Usage:
      INTEGER N
      DOUBLE PRECISION  ANS, A(N), WK(N)
      ANS = DMEDF (A, N, WK)

*Arguments:
      A  :IN      Array of input values.
      N  :IN      Number of elements in A.
      WK:WORK     Work array of size N.

*Function Return Values:
      ANS : the median of the values in A.

*Description:
      DMEDF calculates the median of the N values contained in A. If N
      is odd, the median is the (N + 1)/2 ordered value. For N even,
      the value is the average of the N/2 and N/2 + 1 ordered values.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
              (The above is the date of LCSD-442,Rev.1 and is undoubtedly
              a significant underestimate of the age of this routine.)
      890223  Added SLATEC/LDOC prologue. (FNF)
      890518  Modified sequence numbers to fit in columns 73-80. (FNF)
      920319  Updated with prologue edited 891025 by G. Shaw for manual.
      930930  Converted old UNICOS names to S- or I-names. (DBP)
      931004  Corrected name conversion errors. (FNF)
      931005  Corrected list of equivalent routines, made sure that all
              variables are declared, and improved comments. (FNF)
***  END PROLOGUE  DMEDF
```

DMINAF

```
DOUBLE PRECISION FUNCTION DMINAF (ARRAY, IFIRST, ILAST, ISTRID,
+                                IMIN)
***BEGIN PROLOGUE  DMINAF
***PURPOSE  Minimum value in a one-dimensional array.
***LIBRARY   PMATH
***CATEGORY  N5A
***TYPE      DOUBLE PRECISION (SMINAF-S, DMINAF-D, AMINF8-8, IMINAF-I)
***KEYWORDS  MINIMUM
***AUTHOR  Painter, Jeffrey F., (LLNL)
           Revised by:
           Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine AMINAF.)
*Usage:
    INTEGER  IFIRST, ILAST, ISTRID, IMIN
    DOUBLE PRECISION  ARRAY(n), AMIN, DMINAF
    AMIN = DMINAF (ARRAY, IFIRST, ILAST, ISTRID, IMIN)

*Arguments:
    ARRAY:IN   Real array to be searched.
               n, the dimension of the array, must be no less than
               ILAST.

    IFIRST:IN  First subscript in the array to be searched.

    ILAST :IN  Last subscript in the array to be searched.

    ISTRID:IN  Increment (stride) between successive locations that
               are to be searched.

    IMIN :OUT  Index of the minimum value in the array, i.e., the
               ordinal position of the value in the array.

*Function Return Values:
    AMIN :      Minimum value in the array.

*Description:
    DMINAF finds the minimum value in a one-dimensional real array,
    and returns its index.  In case of multiple minima, the last
    index found is returned.

*Cautions:
    The array is assumed to be subscripted from 1.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
    830401  DATE WRITTEN  (J. F. Painter)
    890502  Added prologue (T. Suyehiro)
    891025  Edited prologue for publication. (G. Shaw)
*** END PROLOGUE  DMINF
```

DMINMX

```
      SUBROUTINE DMINMX (ARRAY, IFIRST, ILAST, ISTRID, AMIN, AMAX,  
+                      IMIN, IMAX)  
***BEGIN PROLOGUE  DMINMX  
***PURPOSE  Minimum and maximum values in a one-dimensional array.  
***LIBRARY   PMATH  
***CATEGORY  N5A  
***TYPE      DOUBLE PRECISION (SMINMX-S, DMINMX-D, AMNMX8-8, IMINMX-I)  
***KEYWORDS  MINIMUM, MAXIMUM  
***AUTHOR   Painter, Jeffrey F., (LLNL)  
            Revised by:  
            Fritsch, Fred N., (LLNL/LC/MSS)  
***DESCRIPTION  
      (Portable version of Cray MATHLIB routine AMINMX.)  
*Usage:  
      INTEGER  IFIRST, ILAST, ISTRID, IMIN, IMAX  
      DOUBLE PRECISION  ARRAY(n), AMIN, AMAX  
      CALL DMINMX (ARRAY, IFIRST, ILAST, ISTRID, AMIN, AMAX,  
*                IMIN, IMAX)  
  
*Arguments:  
      ARRAY:IN   Real array to be searched.  
                 n, the dimension of the array, must be no less than  
                 ILAST.  
  
      IFIRST:IN  First subscript in the array to be searched.  
  
      ILAST :IN   Last subscript in the array to be searched.  
  
      ISTRID:IN   Increment (stride) between successive locations that  
                 are to be searched (>= 1).  
  
      AMIN :OUT   Minimum value in the array.  
  
      AMAX :OUT   Maximum value in the array.  
  
      IMIN :OUT   Index of the minimum value in the array, i.e., the  
                 ordinal position of the value in the array.  
  
      IMAX :OUT   Index of the maximum value in the array, i.e., the  
                 ordinal position of the value in the array.  
  
*Description:  
      DMINMX finds the minimum and maximum values in a one-dimensional  
      real array, and returns their indices.  In case of multiple  
      extrema, the last index found is returned.  
  
      ISTRID should be greater than or equal to 1.  If ISTRID is less  
      than 1, it is assumed to be 1.  
  
*Cautions:  
      The array is assumed to be subscripted from 1.  
***END PROLOGUE  DMINMX
```

DRANF

```
DOUBLE PRECISION FUNCTION DRANF()  
***BEGIN PROLOGUE  DRANF  
***PURPOSE  Uniform random-number generator.  
             The pseudorandom numbers generated by SRANF/DRANF/RANF8  
             are uniformly distributed in the open interval (0,1).  
***LIBRARY   PMATH  
***CATEGORY  L6A21  
***TYPE      DOUBLE PRECISION (SRANF-S, DRANF-D, RANF8-8)  
***KEYWORDS  RANDOM NUMBER GENERATION, UNIFORM DISTRIBUTION  
***AUTHOR    Fritsch, Fred N., (LLNL/LC/MSS)  
             Original CAL version:  
             Margolies, David, (LLNL/USD/MSS)  
             Durst, Mark J. (LLNL/CMRD/SPG)  
***DESCRIPTION  
             (Portable version of Cray MATHLIB routine RANF.)  
*Usage:  
      DOUBLE PRECISION  R, DRANF  
      R = DRANF()  
  
*Function Return Values:  
      R          Random number between 0 and 1.  
  
*Description:  
      DRANF generates pseudorandom numbers lying strictly between 0  
      and 1. Each call to DRANF produces a different value, until the  
      sequence cycles after 2**46 calls.  
  
      DRANF is a linear congruential pseudorandom-number generator.  
      The default starting seed is  
          SEED = 4510112377116321(oct) = 948253fc9cd1(hex).  
      The multiplier is 1207264271730565(oct) = 2875a2e7b175(hex).  
  
*See Also:  
      For exponentially distributed random numbers, use DRLGF instead of  
      DRANF.  
      The starting seed for DRANF may be set via RNSSET.  
      The current DRANF seed may be obtained from RNSGET.  
      The DRANF multiplier may be set via RNMSET (changing the  
      multiplier is not recommended).  
      The number of calls to DRANF may be obtained from RNFCNT.  
  
***ROUTINES CALLED  RANF8  
***REVISION HISTORY  (YYMMDD)  
      800325  DATE WRITTEN  
             (Date from original MATHLIB CAL version.)  
      890421  Added SLATEC/LDOC prologue. (FNF)  
      890530  Minor additions/corrections to prologue. (FNF)  
      891025  Edited prologue for publication. (G. Shaw)  
***END PROLOGUE  DRANF
```

DRANFV

```
SUBROUTINE DRANFV (N, RANOUT)
***BEGIN PROLOGUE  DRANFV
***PURPOSE  Vector uniform random-number generator.
             Returns a vector of numbers from the SRANF/DRANF/RANF8
             sequence.
***LIBRARY    PMATH
***CATEGORY   L6A21
***TYPE       DOUBLE PRECISION (SRANFV-S, DRANFV-D, RANFV8-8)
***KEYWORDS   RANDOM NUMBER GENERATION, UNIFORM DISTRIBUTION, VECTOR
***AUTHOR    Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
             (Portable version of Cray MATHLIB routine RANFV.)
*Usage:
      INTEGER  N
      DOUBLE PRECISION  RANOUT(n)
      CALL DRANFV (N, RANOUT)

*Arguments:
      N      :IN   Number of random numbers to be generated.
      RANOUT:OUT  Vector of N random numbers between 0 and 1.
                  The actual dimension of RANOUT must satisfy n>=N.

*Description:
      DRANFV generates pseudorandom numbers lying strictly between 0
      and 1. The above call is equivalent to the loop
          DO 10 I=1,N
              RANOUT(I) = DRANF()
          10 CONTINUE
      except that DRANFV may be significantly faster for suitable N.
      (The actual timing is likely to be platform-dependent.)

*See Also:
      Refer to DRANF description for information on restarting the
      sequence and related matters.

***ROUTINES CALLED  DRANF
***REVISION HISTORY  (YYMMDD)
      931011  DATE WRITTEN
      931011  Created portable version that merely calls SRANF.  (FNF)
      931018  Produced double precision version.  (DBP)
      931025  Added equivalent routines list.  (FNF)
      940421  Improved purpose.  (FNF)
      940727  Added preprocessor directives for REAL*8 entries.  (FNF)
      951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  DRANFV
```

DRANKS

```
SUBROUTINE DRANKS (A, N, AO, RA, IO, B, ISTAK)
***BEGIN PROLOGUE  DRANKS
***PURPOSE  Ranks of a one-dimensional real array.
***LIBRARY   PMATH
***CATEGORY  L1A
***TYPE      DOUBLE PRECISION (SRANKS-S, DRANKS-D, RANKS8-8)
***KEYWORDS  ELEMENTARY STATISTICS, RANKS
***AUTHOR   Unknown, Name, (LLNL/USD/NMG)
            Currently responsible:
            Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine RANKS.)
*Usage:
    INTEGER  N, IO(N), ISTAK(N)
    DOUBLE PRECISION  A(N), AO(N), RA(N), B(N)
    CALL DRANKS (A, N, AO, RA, IO, B, ISTAK)

*Arguments:
    A      :IN      Array of input values.
    N      :IN      Number of elements in A.
    AO     :OUT     Array containing the values of A ordered.
    RA     :OUT     Array of order N, containing the ranks.
    IO     :WORK    Work array of order N.
    B      :WORK    Work array of order N.
    ISTAK  :WORK    Work array of order N.

*Description:
    DRANKS orders the N values contained in A and calculates their
    ranks.  For ties, the average of the ranks is assigned.

*Accuracy:

*Cautions:
    This routine was formerly known as ORDERS.  Its name was changed
    in March 1991 to avoid conflict with a SCILIB (OMNILIB) routine.

*Portability:

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
    830812  DATE WRITTEN
            (The above is the date of LCSD-442,Rev.1 and is undoubtedly
            a significant underestimate of the age of this routine.)
    890223  Added SLATEC/LDOC prologue. (FNF)
    890518  Modified sequence numbers to fit in columns 73-80. (FNF)
    890518  Changed m from a data-loaded constant to a parameter. (FNF)
***END PROLOGUE  DRANKS
```

DREFIT

```
SUBROUTINE DREFIT (YDATA, NDATA, MTERMS, WEIGHT, COEFF, RSD2,  
+                WORK, JOB, IERR)  
***BEGIN PROLOGUE  DREFIT  
***PURPOSE  Repeated polynomial fitting.  
             SREFIT(DREFIT) is called after a call of SFITPO(DFITPO) to  
             fit a polynomial of the same or lower degree to the same  
             data or to data in which y has been changed but x left the  
             same.  
***LIBRARY    PMATH  
***CATEGORY   K1A1A2, L8B1B1  
***TYPE       DOUBLE PRECISION (SREFIT-S, DREFIT-D, REFIT8-8)  
***KEYWORDS   POLYNOMIAL FITTING, LEAST SQUARES  
***AUTHOR     Painter, Jeffrey F., (LLNL/CMRD)  
             Currently responsible:  
             Fritsch, Fred N., (LLNL/LC/MSS)  
***DESCRIPTION  
             (Portable version of Cray MATHLIB routine REFITP.)  
*Usage:  
    INTEGER  NDATA, MTERMS, JOB, IERR  
    PARAMETER (NWORK = (NDATA+1)*(MTERMS+1) )  
    DOUBLE PRECISION  YDATA(NDATA), WEIGHT(NDATA), COEFF(MTERMS),  
    *                RSD2, WORK(NWORK)  
  
    CALL DREFIT (YDATA, NDATA, MTERMS, WEIGHT, COEFF, RSD2,  
    *                WORK, JOB, IERR)  
  
*Arguments:  
    YDATA :IN   Array of values (new or old) of the dependent  
                variable, y, of dimension NDATA.  
  
    NDATA :IN   Number of data points.  It must be the same as the  
                NDATA used for DFITPO.  
  
    MTERMS:IN   Number of terms in the polynomial to be found.  It  
                cannot be greater than NTERMS, the number of terms  
                in the polynomial that DFITPO found.  
                If MTERMS > NDATA, the result will be the coefficients  
                of an interpolating polynomial of degree NDATA-1, and  
                COEFF(j) = 0 for j > NDATA.  
  
    WEIGHT:IN   Optional weight array.  It must be the same as in  
                the DFITPO call.  
  
    COEFF :OUT  Array containing the MTERMS coefficients of the  
                polynomial.  
  
    RSD2  :OUT  Sum of the squares of the (weighted) residuals  
                corresponding to COEFF.  
  
    WORK :WORK  Must be exactly the same array as in the previous  
                call of DFITPO or DREFIT; no changes may be made by  
                the calling program.  As in DFITPO, WORK contains the  
                residuals R(i) in its first NDATA entries if JOB is  
                nonzero.  
  
    JOB  :IN    Residuals-computation flag:
```


non-0 Residuals are computed and output in WORK.
0 Residuals are not completely computed,
 although RSD2 is computed. (This option will
 more efficient if the R(i) are not required.)

IERR :OUT Error flag. On normal termination, IERR = 0.

Fatal errors:

- (1) DQRSL returned INFO=IERR: 0 < IERR <= NTERMS
A singular matrix has been detected (same meaning
as in DFITPO). DREFIT should not be called if
DFITPO returned IERR > 0.
- (2) MTERMS > NTERMS: IERR = -1
COEFF has not been computed in either case.

*Description:

DREFIT is called, after a call of DFITPO, to fit a polynomial of
the same or lower degree to the same data or to data in which y
has been changed but x left the same as in the DFITPO call.

DREFIT provides the same output as would a second call of DFITPO,
but DREFIT is more efficient. DREFIT may be called any number of
times, as long as the contents of WORK are not disturbed.

*Portability:

This routine calls the LINPACK routine DQRSL, and BLAS (Basic
Linear Algebra Subprograms) DCOPY, DDOT.

*See Also:

See DFITPO for additional information.

***SEE ALSO DFITPO
***REFERENCES (NONE)
***ROUTINES CALLED DCOPY, DDOT, DQRSL
***REVISION HISTORY (YYMMDD)
***END PROLOGUE DREFIT

DRLGF

```
DOUBLE PRECISION FUNCTION DRLGF()  
***BEGIN PROLOGUE  DRLGF  
***PURPOSE  Exponential random-number generator.  
             The pseudorandom numbers generated by SRLGF/DRLGF/RLGF8  
             are drawn from the exponential distribution with mean 1.  
***LIBRARY  PMATH  
***CATEGORY  L6A5  
***TYPE      DOUBLE PRECISION (SRLGF-S, DRLGF-D, RLGF8-8)  
***KEYWORDS  RANDOM NUMBER GENERATION, EXPONENTIAL DISTRIBUTION  
***AUTHOR  Fritsch, Fred N., (LLNL/LC/MSS)  
            Original CAL version:  
            Margolies, David, (LLNL/USD/MSS)  
            Durst, Mark J. (LLNL/CMRD/SPG)  
***DESCRIPTION  
            (Portable version of Cray MATHLIB routine RLGF.)  
*Usage:  
      DOUBLE PRECISION  R  
      R = DRLGF()  
  
*Function Return Values:  
      R      A random number drawn from the exponential distribution  
            with mean 1.  
  
*Description:  
      DRLGF takes the natural logarithm of uniform random numbers.  
      DRLGF() should be used in place of the expression -LOG(DRANF()).  
      Each call to DRLGF produces a different value, until the sequence  
      cycles after 2**46 calls.  
  
      DRLGF uses a linear congruential pseudorandom-number generator  
      which is identical to DRANF except that the default starting seed  
      is different:  
          SEED = 7315512527213717(oct) = ecda555d17cf(hex).  
      The multiplier is 1207264271730565(oct) = 2875a2e7b175(hex).  
      The SRLGF/DRLGF/RLGF8 sequence is independent of that generated  
      by SRANF/DRANF/RANF8.  
  
*Cautions:  
      Note that if you are using both DRANF and DRLGF, stopping and  
      restarting both sequences will require calling both RNSGET/RNSSET  
      and RLSGET/RLSSET.  
  
***ROUTINES CALLED  RLGF8  
***REVISION HISTORY  (YYMMDD)  
***END PROLOGUE  DRLGF
```

DSRCOM

```
      SUBROUTINE DSRCOM (RSAV, ISAV, JOB)
***BEGIN PROLOGUE  DSRCOM
***PURPOSE  Save/restore ODEPACK COMMON blocks.
***LIBRARY  PMATH (ODEPACK)
***CATEGORY  I1C
***TYPE      DOUBLE PRECISION (SSRCOM-S, DSRCOM-D, SRCOM8-8)
***AUTHOR  Hindmarsh, Alan C., (LLNL)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine SRCOM.)

This routine saves or restores (depending on JOB) the contents of
the COMMON block DLS001, which is used internally
by one or more ODEPACK solvers.

RSAV = real array of length 218 or more.
ISAV = integer array of length 37 or more.
JOB   = flag indicating to save or restore the COMMON blocks:
        JOB = 1 if COMMON is to be saved (written to RSAV/ISAV)
        JOB = 2 if COMMON is to be restored (read from RSAV/ISAV)
        A call with JOB = 2 presumes a prior call with JOB = 1.

***SEE ALSO  DLSODE
***ROUTINES CALLED  (NONE)
***COMMON BLOCKS  DLS001
***REVISION HISTORY  (YYMMDD)
    791129  DATE WRITTEN
    890501  Modified prologue to SLATEC/LDOC format.  (FNF)
    890503  Minor cosmetic changes.  (FNF)
    921116  Deleted treatment of block /EH0001/.  (ACH)
    930801  Reduced Common block length by 2.  (ACH)
    930809  Renamed to allow single/double precision versions.  (ACH)
    940315  Added REAL*8 name to C***TYPE line.  (FNF)
    940727  Added preprocessor directives for REAL*8 entries.  (FNF)
    941011  Changed to user-callable.  (FNF)
    951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  DSRCOM
```

DSTDEV

```
      DOUBLE PRECISION FUNCTION DSTDEV (A, N, IND)
***BEGIN PROLOGUE  DSTDEV
***PURPOSE  Standard deviation of a one-dimensional real array.
***LIBRARY   PMATH
***CATEGORY  L1A
***TYPE      DOUBLE PRECISION (SSTDEV-S, DSTDEV-D, STDEV8-8)
***KEYWORDS  ELEMENTARY STATISTICS, STANDARD DEVIATION
***AUTHOR   Unknown, Name (LLNL/USD/NMG)
             Durst, Mark J. (LLNL/CMRD/SPG)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine STDEVF.)
*Usage:
      INTEGER  N, IND
      DOUBLE PRECISION  ANS, A(N)
      ANS = DSTDEV (A, N, IND)

*Arguments:
      A  :IN  Array of input values.
      N  :IN  Number of elements in A.
      IND:IN  Job-control flag:
               0  Divide the adjusted sum of squares by N - 1,
                   producing the usual standard-deviation calculation.
              non-0 Divide by N.

*Function Return Values:
      ANS      The standard deviation of the values in A.

*Description:
      DSTDEV calculates the standard deviation of the N values contained
      in A, as modified by IND.

*See Also:
      For a vector of standard deviations, see DCOVAR.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
              (The above is the date of LCSD-442,Rev.1 and is undoubtedly
              a significant underestimate of the age of this routine.)
      890131  Replaced abs with max1 in argument to sqrt. (FNF)
      890223  Added SLATEC/LDOC prologue. (FNF)
      890518  Modified sequence numbers to fit in columns 73-80. (FNF)
      920319  Updated with prologue edited 891025 by G. Shaw for manual.
***END PROLOGUE  DSTDEV
```

DUMACH

```
DOUBLE PRECISION FUNCTION DUMACH ()
***BEGIN PROLOGUE  DUMACH
***PURPOSE  Compute the unit roundoff of the machine.
***LIBRARY   PMATH
***CATEGORY  R1
***TYPE      DOUBLE PRECISION (RUMACH-S, DUMACH-D, UMACH8-8)
***KEYWORDS  MACHINE CONSTANTS
***AUTHOR   Hindmarsh, Alan C., (LLNL)
***DESCRIPTION
    *Usage:
        DOUBLE PRECISION  A, DUMACH
        A = DUMACH()

    *Function Return Values:
        A : the unit roundoff of the machine.

    *Description:
        The unit roundoff is defined as the smallest positive machine
        number u such that 1.0 + u .ne. 1.0. This is computed by DUMACH
        in a machine-independent manner.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
    930216  DATE WRITTEN
    930818  Added SLATEC-format prologue.  (FNF)
    931026  Minor change to reduce single/double differences.  (FNF)
    940315  Added REAL*8 name to C***TYPE line.  (FNF)
    940727  Added preprocessor directives for REAL*8 entries.  (FNF)
    951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  DUMACH
```

DZERO

```
SUBROUTINE DZERO (F, B, C, ABSERR, RELERR, IFLAG)
***BEGIN PROLOGUE  DZERO
***PURPOSE  Find a root x of a nonlinear equation  $F(x) = 0$ .
             A search interval (b,c) must be supplied such that
              $F(b)*F(c) \leq 0$ .
***LIBRARY    PMATH
***CATEGORY   F1B
***TYPE       DOUBLE PRECISION (SZERO-S, DZERO-D, ZERO8-8)
***KEYWORDS   ZEROFINDING, NONLINEAR EQUATIONS, SECANT METHOD,
             BISECTION METHOD
***AUTHOR    Leonard, L. J., (LLNL)
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
             (Portable version of Cray MATHLIB routine ZEROIN.)
*Usage:
    INTEGER  IFLAG
    DOUBLE PRECISION  F, B, C, ABSERR, RELERR
    EXTERNAL  F
    CALL DZERO (F, B, C, ABSERR, RELERR, IFLAG)

*Arguments:
    F :EXT      Name of a function subprogram defining a continuous
                real function of a single real variable x. The
                calling program must declare the function to be
                EXTERNAL.

    B :INOUT    Input:  Lower bound of the search interval (B,C).
                Output: The better approximation to a root, for B
                and C are redefined so that
                 $ABS(F(B)) \leq ABS(F(C))$ .

    C :INOUT    Input:  Upper bound of the search interval (B,C).
                Output: The value of C is not necessarily close to
                B and should be disregarded (see B above).

    ABSERR:IN   Roughly the maximum difference allowed between B
                and C.  If zero is a possible root, do not use
                ABSERR = 0.

    RELERR:IN   Roughly the maximum relative error allowed between
                B and C; i.e., the degree of accuracy required in
                the root.

    IFLAG:INOUT Input:
                 $\geq 6$   The maximum number of function evaluations
                allowed.
                 $< 6$   The maximum number of evaluations is 100.

                Output:
                1   $F(B) * F(C) < 0$ , and the stopping criterion
                    $ABS(B - C) \leq 2.0 * (RELERR * ABS(B) + ABSERR)$ 
                   is met.
                2  B is found such that  $F(B) = 0$ . The interval
                   (B,C) may or may not have satisfied the stopping
                   criterion.
                3   $ABS(F(B))$  exceeds the absolute values of the
```

- function at the original input values of B and C;
i.e., the values found by DZERO are "worse" than
those supplied in the call. In this case, it is
likely that B is near a pole of the function.
- 4 No odd-order zero was found in the interval. A
local minimum may have been obtained.
 - 5 The stopping criterion is not met within the
specified number of function evaluations.

***Description:**

DZERO finds a root x of the nonlinear equation $F(x) = 0$. Normal input consists of a continuous function F and an initial search interval (B,C) that brackets the desired zero of F ; i.e., $F(B) * F(C) \leq 0$.

Each iteration finds new values of B and C such that the interval (B,C) is shrunk, and $F(B) * F(C) \leq 0$. The stopping criterion is

$$\text{ABS}(B - C) \leq 2.0 * (\text{RELERR} * \text{ABS}(B) + \text{ABSERR})$$

DZERO is a slightly modified version of the subroutine DZERO by Shampine and Allen (see Ref. 2). The method used is a combination of bisection and the secant iteration.

***Cautions:**

F is assumed to be a continuous real-valued function. The algorithm in DZERO assumes that F has exactly one zero in the interval $[B,C]$. If, in fact, F has an odd number of zeros, DZERO will zero in on one of them, giving no indication that there may be more.

***See Also:**

Another implementation of this algorithm may be found in routine FZERO in the SLATEC Common Mathematical Library.

***END PROLOGUE DZERO

IMAXAF

```
      INTEGER FUNCTION IMAXAF (IARRAY, IFIRST, ILAST, ISTRID, IMAX)
***BEGIN PROLOGUE  IMAXAF
***PURPOSE  Maximum value in a one-dimensional array.
***LIBRARY  PMATH
***CATEGORY  N5A
***TYPE      INTEGER (SMAXAF-S, DMAXAF-D, AMAXF8-8, IMAXAF-I)
***KEYWORDS  MAXIMUM
***AUTHOR  Painter, Jeffrey F., (LLNL)
           Revised by:
           Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine MAXAF.)
*Usage:
      INTEGER  IFIRST, ILAST, ISTRID, IMAX
      INTEGER  IARRAY(n), IAMAX, IMAXAF
      IAMAX = IMAXAF (IARRAY, IFIRST, ILAST, ISTRID, IMAX)

*Arguments:
      IARRAY:IN   Integer array to be searched.
                  n, the dimension of the array, must be no less than
                  ILAST.

      IFIRST:IN   First subscript in the array to be searched.

      ILAST :IN   Last subscript in the array to be searched.

      ISTRID:IN   Increment (stride) between successive locations that
                  are to be searched.

      IMAX :OUT   Index of the maximum value in the array, i.e., the
                  ordinal position of the value in the array.

*Function Return Values:
      IAMAX :     Maximum value in the array.

*Description:
      IMAXAF finds the maximum value in a one-dimensional integer array,
      and returns its index.  In case of multiple maxima, the last
      index found is returned.

*Cautions:
      The array is assumed to be subscripted from 1.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830401  DATE WRITTEN  (J. F. Painter)
***END PROLOGUE  IMAXAF
```


IMINAF

```
      INTEGER FUNCTION IMINAF (IARRAY, IFIRST, ILAST, ISTRID, IMIN)
***BEGIN PROLOGUE  IMINAF
***PURPOSE  Minimum value in a one-dimensional array.
***LIBRARY   PMATH
***CATEGORY  N5A
***TYPE      INTEGER (SMINAF-S, DMINAF-D, AMINF8-8, IMINAF-I)
***KEYWORDS  MINIMUM
***AUTHOR   Painter, Jeffrey F., (LLNL)
            Revised by:
            Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine MINAF.)
*Usage:
      INTEGER  IFIRST, ILAST, ISTRID, IMIN
      INTEGER  IARRAY(n), IAMIN, IMINAF
      IAMIN = IMINAF (IARRAY, IFIRST, ILAST, ISTRID, IMIN)

*Arguments:
      IARRAY:IN   Integer array to be searched.
                  n, the dimension of the array, must be no less than
                  ILAST.

      IFIRST:IN   First subscript in the array to be searched.

      ILAST :IN   Last subscript in the array to be searched.

      ISTRID:IN   Increment (stride) between successive locations that
                  are to be searched.

      IMIN :OUT   Index of the minimum value in the array, i.e., the
                  ordinal position of the value in the array.

*Function Return Values:
      IAMIN :     Minimum value in the array.

*Description:
      IMINAF finds the minimum value in a one-dimensional integer array,
      and returns its index.  In case of multiple minima, the last
      index found is returned.

*Cautions:
      The array is assumed to be subscripted from 1.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830401  DATE WRITTEN  (J. F. Painter)
      890502  Added prologue (T. Suyehiro)
      891025  Edited prologue for publication. (G. Shaw)
***END PROLOGUE  IMINAF
```

IMINMX

```
      SUBROUTINE IMINMX (IARRAY, IFIRST, ILAST, ISTRID, IAMIN, IAMAX,  
+                      IMIN, IMAX)  
***BEGIN PROLOGUE  IMINMX  
***PURPOSE  Minimum and maximum values in a one-dimensional array.  
***LIBRARY   PMATH  
***CATEGORY  N5A  
***TYPE      INTEGER (SMINMX-S, DMINMX-D, AMNMX8-8, IMINMX-I)  
***KEYWORDS  MINIMUM, MAXIMUM  
***AUTHOR   Painter, Jeffrey F., (LLNL)  
            Revised by:  
            Fritsch, Fred N., (LLNL/LC/MSS)  
***DESCRIPTION  
      (Portable version of Cray MATHLIB routine MINMX.)  
*Usage:  
      INTEGER  IFIRST, ILAST, ISTRID, IMIN, IMAX  
      INTEGER  IARRAY(n), IAMIN, IAMAX  
      CALL  IMINMX (IARRAY, IFIRST, ILAST, ISTRID, IAMIN, IAMAX,  
*              IMIN, IMAX)  
  
*Arguments:  
      IARRAY:IN   Integer array to be searched.  
                  n, the dimension of the array, must be no less than  
                  ILAST.  
  
      IFIRST:IN   First subscript in the array to be searched.  
  
      ILAST :IN   Last subscript in the array to be searched.  
  
      ISTRID:IN   Increment (stride) between successive locations that  
                  are to be searched (>= 1).  
  
      IAMIN :OUT  Minimum value in the array.  
  
      IAMAX :OUT  Maximum value in the array.  
  
      IMIN :OUT  Index of the minimum value in the array, i.e., the  
                  ordinal position of the value in the array.  
  
      IMAX :OUT  Index of the maximum value in the array, i.e., the  
                  ordinal position of the value in the array.  
  
*Description:  
      IMINMX finds the minimum and maximum values in a one-dimensional  
      integer array, and returns their indices.  In case of multiple  
      extrema, the last index found is returned.  
  
      ISTRID should be greater than or equal to 1.  If ISTRID is less  
      than 1, it is assumed to be 1.  
  
*Cautions:  
      The array is assumed to be subscripted from 1.  
***END PROLOGUE  IMINMX
```

IUMACH

```
      INTEGER FUNCTION IUMACH()  
***BEGIN PROLOGUE  IUMACH  
***PURPOSE  Provide standard output unit number.  
***LIBRARY   PMATH  
***CATEGORY  R1  
***TYPE      INTEGER (IUMACH-I)  
***KEYWORDS  MACHINE CONSTANTS  
***AUTHOR   Hindmarsh, Alan C., (LLNL)  
***DESCRIPTION  
  *Usage:  
      INTEGER  LOUT, IUMACH  
      LOUT = IUMACH()  
  
  *Function Return Values:  
      LOUT : the standard logical unit for Fortran output.  
  
***REFERENCES  (NONE)  
***ROUTINES CALLED  (NONE)  
***REVISION HISTORY  (YYMMDD)  
    930915  DATE WRITTEN  
    930922  Made user-callable, and other cosmetic changes. (FNF)  
    951010  Corrected LIBRARY line. (FNF)  
***END PROLOGUE  IUMACH
```

LDFD

```
      INTEGER FUNCTION LDFD (X, T, N)
***BEGIN PROLOGUE  LDFD
***PURPOSE  Table look-down: locate a value in a decreasing table.
***LIBRARY   PMATH
***CATEGORY  N5B
***TYPE      DOUBLE PRECISION (LDFS-S, LDFD-D, LDF8-8)
***KEYWORDS  TABLE LOOK-UP
***AUTHOR   Dubois, Paul F., (LLNL)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine LDF.)
*Usage:
      INTEGER  INDEX, LDFD, N
      DOUBLE PRECISION  X, T(N)
      INDEX = LDFD (X, T, N)

*Arguments:
      X :IN   Any real number.
      T :IN   Array of N strictly decreasing values (the table).
      N :IN   Length of array T.

*Function Return Values:
      INDEX   The index of the first element in array T that is less
              than or equal to X.  Possible values are:

              INDEX = 1,      if X >= T(1) or N <= 0;
              1 < INDEX <= N, if T(INDEX-1) > X >= T(INDEX);
              INDEX = N + 1,  if X < T(N).

*Description:
      LDFD locates a value between elements of a decreasing table.

***REFERENCES  P. F. Dubois, "Swimming upstream: Calculating table
               lookups and piecewise functions," in Parallel Computa-
               tions, G. Rodrigue, Ed., (Academic Press, New York,
               1982), pp.129-151.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
      890509  Added prologue.  (TS/FNF)
      890511  Augmented reference.  (FNF)
      891025  Edited prologue for publication.  (G. Shaw)
      930727  Corrected SLATEC-format prologue.  (FNF)
      930930  Converted old UNICOS names to S- or I-names.  (DBP)
      931005  Added list of equivalent routines and made sure that all
               variables (and the function itself) are declared.  (FNF)
***END PROLOGUE  LDFD
```

LDFS

```
      INTEGER FUNCTION LDFS (X, T, N)
***BEGIN PROLOGUE  LDFS
***PURPOSE  Table look-down: locate a value in a decreasing table.
***LIBRARY   PMATH
***CATEGORY  N5B
***TYPE      SINGLE PRECISION (LDFS-S, LDFD-D, LDF8-8)
***KEYWORDS  TABLE LOOK-UP
***AUTHOR   Dubois, Paul F., (LLNL)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine LDF.)
*Usage:
      INTEGER  INDEX, LDFS, N
      REAL  X, T(N)
      INDEX = LDFS (X, T, N)

*Arguments:
      X :IN   Any real number.
      T :IN   Array of N strictly decreasing values (the table).
      N :IN   Length of array T.

*Function Return Values:
      INDEX   The index of the first element in array T that is less
              than or equal to X.  Possible values are:

              INDEX = 1,      if X >= T(1) or N <= 0;
              1 < INDEX <= N, if T(INDEX-1) > X >= T(INDEX);
              INDEX = N + 1,  if X < T(N).

*Description:
      LDFS locates a value between elements of a decreasing table.

***REFERENCES  P. F. Dubois, "Swimming upstream: Calculating table
               lookups and piecewise functions," in Parallel Computa-
               tions, G. Rodrigue, Ed., (Academic Press, New York,
               1982), pp.129-151.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
      890509  Added prologue.  (TS/FNF)
      890511  Augmented reference.  (FNF)
      891025  Edited prologue for publication.  (G. Shaw)
      930727  Corrected SLATEC-format prologue.  (FNF)
      930930  Converted old UNICOS names to S- or I-names.  (DBP)
      931005  Added list of equivalent routines and made sure that all
               variables (and the function itself) are declared.  (FNF)
      931014  Changed SLDF to the default integer name LDFS.  (FNF)
      931025  Removed unnecessary Caution.  (FNF)
      940421  Improved category.  (FNF)
      951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  LDFS
```

LUFD

```
      INTEGER FUNCTION LUFD (X, T, NBIG)
***BEGIN PROLOGUE  LUFD
***PURPOSE  Table look-up: locate a value in an increasing table.
***LIBRARY   PMATH
***CATEGORY  N5B
***TYPE      DOUBLE PRECISION (LUFS-S, LUFD-D, LUF8-8)
***KEYWORDS  TABLE LOOK-UP
***AUTHOR   Dubois, Paul F., (LLNL)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine LUF.)
*Usage:
      INTEGER  INDEX, LUFD, N
      DOUBLE PRECISION  X, T(N)
      INDEX = LUFD (X, T, N)

*Arguments:
      X :IN   Any real number.
      T :IN   Array of N strictly increasing values (the table).
      N :IN   Length of array T.

*Function Return Values:
      INDEX   The index of the first element in array T that is greater
              than X.  Possible values are:

              INDEX = 1,      if X < T(1) or N <= 0;
              1 < INDEX <= N, if T(INDEX-1) <= X < T(INDEX);
              INDEX = N + 1,  if X >= T(N).

*Description:
      LUFD locates a value between elements of an increasing table.

***REFERENCES  P. F. Dubois, "Swimming upstream: Calculating table
               lookups and piecewise functions," in Parallel Computa-
               tions, G. Rodrigue, Ed., (Academic Press, New York,
               1982), pp.129-151.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
      890509  Added prologue.  (TS/FNF)
      890511  Augmented reference.  (FNF)
      891025  Edited prologue for publication.  (G. Shaw)
      930727  Corrected SLATEC-format prologue.  (FNF)
      930930  Converted old UNICOS names to S- or I-names.  (DBP)
      931004  Corrected name conversion errors.  (FNF)
***END PROLOGUE  LUFD
```

LUFS

```
      INTEGER FUNCTION LUFS (X, T, NBIG)
***BEGIN PROLOGUE  LUFS
***PURPOSE  Table look-up: locate a value in an increasing table.
***LIBRARY   PMATH
***CATEGORY  N5B
***TYPE      SINGLE PRECISION (LUFS-S, LUFD-D, LUF8-8)
***KEYWORDS  TABLE LOOK-UP
***AUTHOR   Dubois, Paul F., (LLNL)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine LUF.)
*Usage:
      INTEGER  INDEX, LUFS, N
      REAL  X, T(N)
      INDEX = LUFS (X, T, N)

*Arguments:
      X :IN   Any real number.
      T :IN   Array of N strictly increasing values (the table).
      N :IN   Length of array T.

*Function Return Values:
      INDEX   The index of the first element in array T that is greater
              than X.  Possible values are:

              INDEX = 1,      if X < T(1) or N <= 0;
              1 < INDEX <= N, if T(INDEX-1) <= X < T(INDEX);
              INDEX = N + 1,  if X >= T(N).

*Description:
      LUFS locates a value between elements of an increasing table.

***REFERENCES  P. F. Dubois, "Swimming upstream: Calculating table
               lookups and piecewise functions," in Parallel Computa-
               tions, G. Rodrigue, Ed., (Academic Press, New York,
               1982), pp.129-151.
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
      890509  Added prologue.  (TS/FNF)
      890511  Augmented reference.  (FNF)
      891025  Edited prologue for publication. (G. Shaw)
      930727  Corrected SLATEC-format prologue.  (FNF)
      930930  Converted old UNICOS names to S- or I-names.  (DBP)
      931004  Corrected name conversion errors.  (FNF)
***END PROLOGUE  LUFS
```

LUGD

```
      INTEGER FUNCTION LUGD (X, T, N, IG)
111. optimize
***BEGIN PROLOGUE  LUGD
***PURPOSE  Table look-up with guess: locate a value in an increasing
           table.
***LIBRARY    PMATH
***CATEGORY   N5B
***TYPE       DOUBLE PRECISION (LUGS-S, LUGD-D, LUG8-8)
***KEYWORDS   TABLE LOOK-UP, ESTIMATED
***AUTHOR    Dubois, Paul F., (LLNL)
           Currently responsible:
           Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine LUG.)
*Usage:
      INTEGER  INDEX, LUGD, N, IG
      DOUBLE PRECISION  X, T(N)
      INDEX = LUGD (X, T, N, IG)

*Arguments:
      X  :IN      Any real number.
      T  :IN      Array of N strictly increasing values (the table).
      N  :IN      Length of array T.
      IG :INOUT   An estimated value for INDEX.  On return, IG = INDEX.

*Function Return Values:
      INDEX      The index of the first element in array T that is
                greater than X.  Possible values are:

                INDEX = 1,      if X < T(1) or N <= 0;
                1 < INDEX <= N, if T(INDEX-1) <= X < T(INDEX);
                INDEX = N + 1,  if X >= T(N).

*Description:
      LUGD locates a value between elements of an increasing table with
      a guess, IG.  If IG is close to the correct index, LUGD will be
      significantly faster than LUFD, especially on large tables.  This
      routine is particularly useful when looking up a series of nearby
      values.

***REFERENCES  P. F. Dubois, "Swimming upstream: Calculating table
                lookups and piecewise functions," in Parallel Computa-
                tions, G. Rodrigue, Ed., (Academic Press, New York,
                1982), pp.129-151.
***ROUTINES CALLED  LUFD
***REVISION HISTORY  (YYMMDD)
      790629  DATE WRITTEN
***END PROLOGUE  LUGD
```


LUGS

```
      INTEGER FUNCTION LUGS (X, T, N, IG)
111. optimize
***BEGIN PROLOGUE  LUGS
***PURPOSE  Table look-up with guess: locate a value in an increasing
            table.
***LIBRARY    PMATH
***CATEGORY   N5B
***TYPE       SINGLE PRECISION (LUGS-S, LUGD-D, LUG8-8)
***KEYWORDS   TABLE LOOK-UP, ESTIMATED
***AUTHOR    Dubois, Paul F., (LLNL)
            Currently responsible:
            Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
            (Portable version of Cray MATHLIB routine LUG.)
*Usage:
      INTEGER  INDEX, LUGS, N, IG
      REAL  X, T(N)
      INDEX = LUGS (X, T, N, IG)

*Arguments:
      X  :IN      Any real number.
      T  :IN      Array of N strictly increasing values (the table).
      N  :IN      Length of array T.
      IG :INOUT   An estimated value for INDEX.  On return, IG = INDEX.

*Function Return Values:
      INDEX      The index of the first element in array T that is
                greater than X.  Possible values are:

                INDEX = 1,      if X < T(1) or N <= 0;
                1 < INDEX <= N, if T(INDEX-1) <= X < T(INDEX);
                INDEX = N + 1,  if X >= T(N).

*Description:
      LUGS locates a value between elements of an increasing table with
      a guess, IG.  If IG is close to the correct index, LUGS will be
      significantly faster than LUFS, especially on large tables.  This
      routine is particularly useful when looking up a series of nearby
      values.

***REFERENCES  P. F. Dubois, "Swimming upstream: Calculating table
                lookups and piecewise functions," in Parallel Computa-
                tions, G. Rodrigue, Ed., (Academic Press, New York,
                1982), pp.129-151.
***ROUTINES CALLED  LUFS
***REVISION HISTORY  (YYMMDD)
      790629  DATE WRITTEN
            (Above is "date last changed" found in source file.)
***END PROLOGUE  LUGS
```

RANF8

```
***BEGIN PROLOGUE  RANF8
***PURPOSE  Uniform random-number generator.
              The pseudorandom numbers generated by SRANF/DRANF/RANF8
              are uniformly distributed in the open interval (0,1).
***LIBRARY    PMATH
***CATEGORY   L6A21
***TYPE       REAL*8 (SRANF-S, DRANF-D, RANF8-8)
***KEYWORDS   RANDOM NUMBER GENERATION, UNIFORM DISTRIBUTION
***AUTHOR    Rathkopf, Jim, (LLNL/CP-Division)
              Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
              (Portable version of Cray MATHLIB routine RANF.)
*Usage:
      REAL*8  R, RANF8
      R = RANF8()

*Function Return Values:
      R          A random number between 0 and 1.

*Description:
      RANF8 generates pseudorandom numbers lying strictly between 0
      and 1. Each call to RANF8 produces a different value, until the
      sequence cycles after 2**46 calls.

      RANF8 is a linear congruential pseudorandom-number generator.
      The default starting seed is
              SEED = 4510112377116321(oct) = 948253fc9cd1(hex).
      The multiplier is 1207264271730565(oct) = 2875a2e7b175(hex).

*See Also:
      For exponentially distributed random numbers, use RLGF8 instead of
      RANF8.
      The starting seed for RANF8 may be set via RNSSET.
      The current RANF8 seed may be obtained from RNSGET.
      The RANF8 multiplier may be set via RNMSET (changing the
      multiplier is not recommended).
      The number of calls to RANF8 may be obtained from RNFCNT.

*Portability:
      This C routine is contained in pmath_rnf.c, which requires header
      files pm_params.h, pm_cnvset.h, and pm_rnfset.h to set up correct
      Fortran binding.

***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      930308  DATE WRITTEN
***END PROLOGUE  RANF8
```

RLFCNT

```
***BEGIN PROLOGUE  RLFCNT
***PURPOSE  Count the number of calls to RLGF family generators.
***LIBRARY   PMATH
***CATEGORY  L6C
***TYPE      ALL (RLFCNT-A)
***KEYWORDS  RANDOM NUMBER GENERATION, EXPONENTIAL DISTRIBUTION, COUNT
***AUTHOR   Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine RLCNT.)
*Usage:
    INTEGER  NUM
    CALL RLFCNT (NUM)

*Arguments:
    NUM :OUT  Number of calls to RLGF8 since the beginning of the
              program.

*Description:
    RLFCNT returns the number of calls to RLGF8 made since the
    beginning of the program.  This count will also include any calls
    to SRLGF or DRLGF.

*Portability:
    This C routine is contained in pmath_rlf.c, which requires header
    files pm_params.h, pm_cnvset.h, and pm_rlfset.h to set up correct
    Fortran binding.

***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
    931022  DATE WRITTEN
    931122  Changed name from PM_RNFCNT to PM_RLFCNT.  (FNF)
    940425  Added SLATEC-format prologue.  (FNF)
***END PROLOGUE  RLFCNT
```

RLGF8

```
***BEGIN PROLOGUE  RLGF8
***PURPOSE  Exponential random-number generator.
              The pseudorandom numbers generated by SRLGF/DRLGF/RLGF8
              are drawn from the exponential distribution with mean 1.
***LIBRARY    PMATH
***CATEGORY   L6A5
***TYPE       REAL*8 (SRLGF-S, DRLGF-D, RLGF8-8)
***KEYWORDS   RANDOM NUMBER GENERATION, EXPONENTIAL DISTRIBUTION
***AUTHOR    Rathkopf, Jim, (LLNL/CP-Division)
              Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
              (Portable version of Cray MATHLIB routine RLGF.)

*Usage:
      REAL*8  R, RLGF8
      R = RLGF8()

*Function Return Values:
      R          A random number drawn from the exponential distribution
                  with mean 1.

*Description:
      RLGF8 takes the natural logarithm of uniform random numbers.
      RLGF8() should be used in place of the expression -LOG(RANF8()).
      Each call to RLGF8 produces a different value, until the sequence
      cycles after 2**46 calls.

      RLGF8 uses a linear congruential pseudorandom-number generator
      which is identical to RANF8 except that the default starting seed
      is different:
              SEED = 7315512527213717(oct) = ecda555d17cf(hex).
      The multiplier is 1207264271730565(oct) = 2875a2e7b175(hex).
      The SRLGF/DRLGF/RLGF8 sequence is independent of that generated
      by SRANF/DRANF/RANF8.

*Cautions:
      Note that if you are using both RANF8 and RLGF8, stopping and
      restarting both sequences will require calling both RNSGET/RNSSET
      and RLSGET/RLSSET.

*Portability:
      This C routine is contained in pmath_rlf.c, which requires header
      files pm_params.h, pm_cnvset.h, and pm_rlfset.h to set up correct
      Fortran binding.

***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
***END PROLOGUE  RLGF8
```

RLMSET

```
***BEGIN PROLOGUE  RLMSET
***PURPOSE  Set multiplier for RLGF family generators.
***LIBRARY   PMATH
***CATEGORY  L6C
***TYPE      ALL (RLMSET-A)
***KEYWORDS  RANDOM NUMBER GENERATION, EXPONENTIAL DISTRIBUTION,
             MULTIPLIER
***AUTHOR   Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine RLGMSSET.)
*Usage:
    INTEGER  NEWMUL
    CALL RLMSET (NEWMUL)

    or

    INTEGER  NEWMUL
    REAL*8  OLDMUL, RNMUSET
    OLDMUL = RLMSET(NEWMUL)

*Arguments:
    NEWMUL :IN    The new multiplier desired (odd, >1, <2**46).

*Function Return Values:
    OLDMUL = 0      NEWMUL replaced the multiplier.
    = non-0        The old multiplier, if it was not replaced.

    This alternate calling form is intended for use in a statement
    of the form
        IF ( RLMSET(NEWMUL).NE.0 ) GO TO ERROR

*Description:
    RLMSET changes the multiplier used by SRLGF/DRLGF/RLGF8.
    See "Cautions" below!

    NEWMUL must be odd and greater than 1. It must also be less than
    2**46 = 70368744177664. If any of these checks fail, the multiplier
    will not be changed, and a nonzero value is returned. (The
    default multiplier is 1207264271730565(oct) = 2875a2e7b175(hex).)

*Cautions:
    Changing the multiplier is NOT recommended. Most values are poor
    multipliers. A poor multiplier will cause the sequence of pseudo-
    random numbers to have very undesirable statistical properties.

*Portability:
    This C routine is contained in pmath_rlf.c, which requires header
    files pm_params.h, pm_cnvset.h, and pm_rlfset.h to set up correct
    Fortran binding.

*See Also:
    SRLGF/DRLGF/RLGF8 is the exponential random-number generator.
    CV16TO64 may be useful for constructing NEWMUL.
    CV64TO16 may be useful for saving OLDMUL.

***ROUTINES CALLED  CV16TO64, CV64TO16
***REVISION HISTORY  (YMMDD)
    950928  Added return value, as in MATHLIB routine, and corrected
           to not reset to the default value when input argument is
```

```
zero. (FNF)
951002 Replaced union (that doesn't work on the Cray) with coding
      that calls PM_16TO64 or PM_64TO16 (i.e., CV16TO64 or
      CV64TO16). (FNF)
951027 Added upper bound restriction and added checks that the
      input value is an acceptable multiplier. (FNF)
***END PROLOGUE  RLMSET
```

RLSGET

```
***BEGIN PROLOGUE  RLSGET
***PURPOSE  Get seed for RLGF family generators.
***LIBRARY   PMATH
***CATEGORY  L6C
***TYPE      ALL (RLSGET-A)
***KEYWORDS  RANDOM NUMBER GENERATION, EXPONENTIAL DISTRIBUTION, SEED
***AUTHOR   Rathkopf, Jim, (LLNL/CP-Division)
            Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine RLGGET.)
*Usage:
    REAL*8  SEED, RLSGET
    SEED = RLSGET ()

*Function Return Values:
    SEED      The seed after the return from RLSGET.

*Description:
    RLSGET returns the value of the current SRLGF/DRLGF/RLGF8 seed.
    This value can be saved and used with RLSSET to reproduce a
    portion of the SRLGF/DRLGF/RLGF8 sequence.

*Cautions:
    The exact bit pattern of SEED is important.  If SEED is to be used
    to reset the sequence via RLSSET, it should not be modified in any
    way.

    RLSGET is a function of type REAL*8.  This means that both RLSGET
    and SEED should be typed REAL*8, as above.  Otherwise, Fortran's
    implicit type conventions will assume that both SEED and RLSGET
    are type REAL.  This works on the Cray, but not on workstations
    which have 32-bit words, since the seed requires 48 bits.  In any
    case, do not perform any arithmetic with the seed.

*Portability:
    This C routine is contained in pmath_rlf.c, which requires header
    files pm_params.h, pm_cnvset.h, and pm_rlfset.h to set up correct
    Fortran binding.

*See Also:
    SRLGF/DRLGF/RLGF8 is the exponential random-number generator.
    RLSSET changes the value of the SRLGF/DRLGF/RLGF8 seed.
    CV64TO16 may be useful for saving SEED.

***ROUTINES CALLED  CV16TO64
***REVISION HISTORY  (YYMMDD)
    930308  DATE WRITTEN
***END PROLOGUE  RLSGET
```

RLSSET

```
***BEGIN PROLOGUE  RLSSET
***PURPOSE  Set seed for RLGF family generators.
***LIBRARY   PMATH
***CATEGORY  L6C
***TYPE      ALL (RLSSET-A)
***KEYWORDS  RANDOM NUMBER GENERATION, EXPONENTIAL DISTRIBUTION, SEED
***AUTHOR   Rathkopf, Jim, (LLNL/CP-Division)
            Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine RLGSET.)
*Usage:
    REAL*8  SEED
    CALL RLSSET (SEED)

*Arguments:
    SEED :IN    The new seed desired.

*Description:
    RLSSET changes the value of the SRLGF/DRLGF/RLGF8 seed.
    It can be used inconjunction with RLSGET to reproduce a
    portion of the SRLGF/DRLGF/RLGF8 sequence.

    SEED must be a REAL*8 variable.  It should be odd, but if it
    is not, RLSSET makes it odd.  Only the lower 48 bits of SEED
    are used.

    If SEED = 0, the default seed is used:
        SEED = 7315512527213717(oct) = ecda555d17cf(hex).
    that is, the sequence is restarted.

*Cautions:
    The next value of RLGF8 will be  $-\log(\text{SEED} * 2^{*-48})$ .  It is
    recommended to call RLGF8 several times without using the results
    in order to avoid unusually small numbers.

    On workstations, which have 32-bit floating point and often have
    16-bit integer arithmetic by default, some care may be required
    to insure that all bits of SEED are correctly transmitted to
    RLSSET.  This can be accomplished by using CV16TO64 to load it.
    For example, to set the seed to 1234567890ab(hex):
        INTEGER  ISEED(3)
        REAL*8  SEED
        DATA  ISEED /X'1234', X'5678', X'90AB'/
        CALL CV16TO64 (ISEED, SEED)
        CALL RLSSET (SEED)

*Portability:
    This C routine is contained in pmath_rlf.c, which requires header
    files pm_params.h, pm_cnvset.h, and pm_rlfset.h to set up correct
    Fortran binding.

***ROUTINES CALLED  CV64TO16
***REVISION HISTORY  (YYMMDD)
    950927  Replaced rnset call and unnecessary multiplier resetting by
           a direct call to rand48_16to24. (FNF)
    950927  Corrected erroneous default seed when SEED=0.  (Previously
```



```
        it set to the RANF value.) (FNF)
951002  Replaced union (that doesn't work on the Cray) with coding
        that calls PM_64TO16 (i.e., CV64TO16). (FNF)
951027  Implemented check for odd SEED. (FNF)
***END PROLOGUE  RLSSET
```

RNFCNT

```
***BEGIN PROLOGUE  RNFCNT
***PURPOSE  Count the number of calls to RANF family generators.
***LIBRARY   PMATH
***CATEGORY  L6C
***TYPE      ALL (RNFCNT-A)
***KEYWORDS  RANDOM NUMBER GENERATION, UNIFORM DISTRIBUTION, COUNT
***AUTHOR   Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine RNCOUNT.)
*Usage:
    INTEGER  NUM
    CALL RNFCNT (NUM)

*Arguments:
    NUM :OUT  Number of calls to RANF8 since the beginning of the
              program.

*Description:
    RNFCNT returns the number of calls to RANF8 made since the
    beginning of the program.  This count will also include any calls
    to SRANF or DRANF.

*Portability:
    This C routine is contained in pmath_rnf.c, which requires header
    files pm_params.h, pm_cnvset.h, and pm_rnfset.h to set up correct
    Fortran binding.

***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
    931022  DATE WRITTEN
    940425  Added SLATEC-format prologue.  (FNF)
***END PROLOGUE  RNFCNT
```

RNMSET

```
***BEGIN PROLOGUE  RNMSET
***PURPOSE  Set multiplier for RANF family generators.
***LIBRARY   PMATH
***CATEGORY  L6C
***TYPE      ALL (RNMSET-A)
***KEYWORDS  RANDOM NUMBER GENERATION, UNIFORM DISTRIBUTION, MULTIPLIER
***AUTHOR   Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine RNMUSET.)
*Usage:
    INTEGER  NEWMUL
    CALL RNMSET (NEWMUL)

or

    INTEGER  NEWMUL
    REAL*8  OLDMUL, RNMUSET
    OLDMUL = RNMSET(NEWMUL)

*Arguments:
    NEWMUL :IN    The new multiplier desired (odd, >1, <2**46).

*Function Return Values:
    OLDMUL = 0      NEWMUL replaced the multiplier.
             = non-0  The old multiplier, if it was not replaced.

    This alternate calling form is intended for use in a statement
    of the form
        IF ( RNMSET(NEWMUL).NE.0 ) GO TO ERROR

*Description:
    RNMSET changes the multiplier used by SRANF/DRANF/RANF8.
    See "Cautions" below!

    NEWMUL must be odd and greater than 1. It must also be less than
    2**46 = 70368744177664. If any of these checks fail, the multiplier
    will not be changed, and a nonzero value is returned. (The
    default multiplier is 1207264271730565(oct) = 2875a2e7b175(hex).)

*Cautions:
    Changing the multiplier is NOT recommended. Most values are poor
    multipliers. A poor multiplier will cause the sequence of pseudo-
    random numbers to have very undesirable statistical properties.
    If the actual value of the multiplier is desired, it and RNMSET
    must be typed REAL*8, as indicated above, so that the result is
    large enough to hold a 48-bit integer. OLDMUL must not be changed
    in any way if it is to be used in a subsequent RNMSET call.

*Portability:
    This C routine is contained in pmath_rnf.c, which requires header
    files pm_params.h, pm_cnvset.h, and pm_rnfset.h to set up correct
    Fortran binding.

***ROUTINES CALLED  CV16TO64, CV64TO16
***REVISION HISTORY  (YYMMDD)
    931109  DATE WRITTEN
             (Created from PM_RNSSET.)
    931215  Reversed order of bytes in multiplier to agree with Cray.  (FNF)
```

940425 Added SLATEC-format prologue. (FNF)
950913 Added conditional-compile blocks to not reverse byte order
if on DEC. (FNF)
950927 Replaced rnset call and unnecessary seed resetting by a
direct call to rand48_16to24. (FNF)
950928 Added return value, as in MATHLIB routine, and corrected
to not reset to the default value when input argument is
zero. (FNF)
951002 Replaced union (that doesn't work on the Cray) with coding
that calls PM_16TO64 or PM_64TO16 (i.e., CV16TO64 or
CV64TO16). (FNF)
951027 Added upper bound restriction and added checks that the
input value is an acceptable multiplier. (FNF)
***END PROLOGUE RNMSET

RNSGET

```
***BEGIN PROLOGUE  RNSGET
***PURPOSE  Get seed for RANF family generators.
***LIBRARY   PMATH
***CATEGORY  L6C
***TYPE      ALL (RNSGET-A)
***KEYWORDS  RANDOM NUMBER GENERATION, UNIFORM DISTRIBUTION, SEED
***AUTHOR   Rathkopf, Jim, (LLNL/CP-Division)
            Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine RANGET.)
*Usage:
    REAL*8  SEED, RNSGET
    SEED = RNSGET ()

*Function Return Values:
    SEED      The seed after the return from RNSGET.

*Description:
    RNSGET returns the value of the current SRANF/DRANF/RANF8 seed.
    This value can be saved and used with RNSSET to reproduce a
    portion of the SRANF/DRANF/RANF8 sequence.

*Cautions:
    The exact bit pattern of SEED is important.  If SEED is to be used
    to reset the sequence via RNSSET, it should not be modified in any
    way.

    RNSGET is a function of type REAL*8.  This means that both RNSGET
    and SEED should be typed REAL*8, as above.  Otherwise, Fortran's
    implicit type conventions will assume that both SEED and RNSGET
    are type REAL.  This works on the Cray, but not on workstations
    which have 32-bit words, since the seed requires 48 bits.  In any
    case, do not perform any arithmetic with the seed.

*Portability:
    This C routine is contained in pmath_rnf.c, which requires header
    files pm_params.h, pm_cnvset.h, and pm_rnfset.h to set up correct
    Fortran binding.

*See Also:
    SRANF/DRANF/RANF8 is the basic uniform random-number generator.
    RNSSET changes the value of the SRANF/DRANF/RANF8 seed.
    CV64TO16 may be useful for saving SEED.

***ROUTINES CALLED  CV16TO64
***REVISION HISTORY  (YYMMDD)
    930308  DATE WRITTEN
***END PROLOGUE  RNSGET
```

RNSSET

```
***BEGIN PROLOGUE  RNSSET
***PURPOSE  Set seed for RANF family generators.
***LIBRARY   PMATH
***CATEGORY  L6C
***TYPE      ALL (RNSSET-A)
***KEYWORDS  RANDOM NUMBER GENERATION, UNIFORM DISTRIBUTION, SEED
***AUTHOR   Rathkopf, Jim, (LLNL/CP-Division)
            Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine RANSET.)
*Usage:
    REAL*8  SEED
    CALL RNSSET (SEED)

*Arguments:
    SEED :IN   The new seed desired.

*Description:
    RNSSET changes the value of the SRANF/DRANF/RANF8 seed.
    It can be used in conjunction with RNSGET to reproduce a
    portion of the SRANF/DRANF/RANF8 sequence.

    SEED must be a REAL*8 variable.  It should be odd, but if it
    is not, RNSSET makes it odd.  Only the lower 48 bits of SEED
    are used.

    If SEED = 0, the default seed is used:
        SEED = 4510112377116321(oct) = 948253fc9cd1(hex);
    that is, the sequence is restarted.

*Cautions:
    The next value of RANF8 will be SEED * 2**48.  It is recommended
    to call RANF8 several times without using the results in order to
    avoid unusually small numbers.

*Portability:
    This C routine is contained in pmath_rnf.c, which requires
    header files pm_params.h, pm_cnvset.h, and pm_rnfset.h to set up
    corrent Fortran binding.

***ROUTINES CALLED  CV64TO16
***REVISION HISTORY  (YYMMDD)
    930308  DATE WRITTEN
            (Date from Biester's math_rnf.c.)
***END PROLOGUE  RNSSET
```

RUMACH

```
      REAL FUNCTION RUMACH ()
***BEGIN PROLOGUE  RUMACH
***PURPOSE  Compute the unit roundoff of the machine.
***LIBRARY   PMATH
***CATEGORY  R1
***TYPE      SINGLE PRECISION (RUMACH-S, DUMACH-D, UMACH8-8)
***KEYWORDS  MACHINE CONSTANTS
***AUTHOR   Hindmarsh, Alan C., (LLNL)
***DESCRIPTION
  *Usage:
      REAL  A, RUMACH
      A = RUMACH()

  *Function Return Values:
      A : the unit roundoff of the machine.

  *Description:
      The unit roundoff is defined as the smallest positive machine
      number u such that 1.0 + u .ne. 1.0. This is computed by RUMACH
      in a machine-independent manner.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
  930216  DATE WRITTEN
  930818  Added SLATEC-format prologue.  (FNF)
  940315  Added REAL*8 name to C***TYPE line.  (FNF)
  940727  Added preprocessor directives for REAL*8 entries.  (FNF)
  951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  RUMACH
```

SCONST

```
REAL FUNCTION SCONST (NAME)
***BEGIN PROLOGUE  SCONST
***PURPOSE  Provides values for common mathematical constants.
***LIBRARY   PMATH
***CATEGORY  R1
***TYPE      SINGLE PRECISION (SCONST-S, DCONST-D, CONST8-8)
***KEYWORDS  CONSTANTS, PI, TWOPI, PI180, PI3, TWOPI3, FOURPI3, UROUND,
              ONE3, ONE27
***AUTHOR    Basinger, R.C., (LLNL/CMRD)
              Currently responsible:
              Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine CONSTANT.)
*Usage:
    CHARACTER*n  NAME
    REAL  VALUE, SCONST

    NAME = 'name'
    VALUE = SCONST (NAME)
or
    VALUE = SCONST ('name')
```

*Arguments:

NAME :IN Name of the desired constant. Valid names and their meanings are:

I	Name	Value	Meaning
-	-----	-----	-----
1	'pi'	pi	PI = 4.0*ATAN(1.0)
2	'twopi'	2pi	2.0*PI
3	'pi180'	pi/180	PI/180.0
4	'pi3'	pi/3	PI/3.0
5	'twopi3'	2pi/3	2.0*PI/3.0
6	'fourpi3'	4pi/3	4.0*PI/3.0
7	'uround'	unit roundoff	The smallest positive floating- point number such that 1.0 + 'uround' .NE. 1.0
8	'one3'	1/3	1.0/3.0
9	'one27'	1/27	1.0/27.0

Here "pi" in the Value column represents the Greek letter pi, the standard notation for the ratio of the circumference to the diameter of a circle.

The name of the constant may be given in either upper or lower case (but not mixed case).

*Function Return Values:

VALUE : the value of the named constant.

*Description:

SCONST provides values for commonly used mathematical constants. This provides a machine-independent way to obtain correct values for these constants.

*Accuracy:

All values except for element 7 are data-loaded with 32-digit decimal constants generated using Macsyma. We rely on the compiler generating correctly rounded machine values from them. SCONST('uround') is obtained from RUMACH.

*Cautions:

The present version terminates with a STOP statement if NAME is not a valid name.

***REFERENCES (NONE)

***ROUTINES CALLED RUMACH

***REVISION HISTORY (YYMMDD)

820514 DATE WRITTEN

(The above is the date found in the source code. It may be an underestimate of the age of this routine.)

890224 Added SLATEC/LDOC prologue. (FNF)

890301 Made changes to comments per feedback from Tok. (FNF)

890301 Replaced double quote (") as string delimiter in DATA statements with the ANSI standard single quote ('). (FNF)

900627 Changed hexadecimal constants from CIVIC to CFT77 form. (FNF)

920313 Made minor cosmetic changes and changed DATA-loaded value of N to the actual number of available constants. (FNF)

920316 Modified to recognize either upper or lower case names.

Removed the common blocks in the process. (FNF)

920319 Updated with prologue edited 891025 by G. Shaw for manual.

930823 1. Replaced calls to BASELIB routine ZVSEEK with a loop.

2. Rearranged DATA statements to facilitate subsequent changes. (FNF)

930824 Changed names from INTEGER to the more standard CHARACTER type. (FNF)

930826 Eliminated distinction between N, the number of constants, and the dimensions of the arrays. (FNF)

***END PROLOGUE SCONST

SCORRV

```
      SUBROUTINE SCORRV (VCV, M, WK)
***BEGIN PROLOGUE  SCORRV
***PURPOSE  Calculate the correlation matrix from the variance-
             covariance matrix.
***LIBRARY   PMATH
***CATEGORY  L1B
***TYPE      SINGLE PRECISION (SCORRV-S, DCORRV-D, CORRV8-8)
***KEYWORDS  ELEMENTARY STATISTICS, CORRELATION MATRIX
***AUTHOR    Unknown, Name (LLNL/USD/NMG)
             Durst, Mark J. (LLNL/CMRD/SPG)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine CORR.V.)
*Usage:
      INTEGER  M
      PARAMETER (nvcv = (M*(M+1))/2)
      REAL  VCV(nvcv), WK(M)
      CALL SCORRV (VCV, M, WK)

*Arguments:
      VCV:INOUT  Input:  Array of order M(M + 1)/2 containing the
                   variance-covariance matrix in symmetric storage mode.
                   Output: Array containing the correlation matrix in
                   symmetric storage mode.
      M :IN      Number of variables for which correlations are
                   calculated.
      WK :WORK    Work array of order M.

*Description:
      SCORRV calculates the correlation matrix from the variance-
      covariance matrix stored in VCV in symmetric storage mode.  The
      correlation matrix will replace VCV on return.

      "Symmetric storage mode" means (S is taken to be the full matrix):

      VCV(k) = S(i,j),    k = (i(i - 1))/2 + j, i = 1,...,M, j <= i

*See Also:
      SCORRV can be used in conjunction with SCOVAR to obtain both the
      variance-covariance and correlation matrices.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
      931005  Augmented list of equivalent routines, made sure that all
              variables are declared, and improved comments.  (FNF)
      940727  Added preprocessor directives for REAL*8 entries.  (FNF)
      951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  SCORRV
```

SCOVAR

```
SUBROUTINE SCOVAR (A, N, M, IND, VCV, SD, WK)
***BEGIN PROLOGUE  SCOVAR
***PURPOSE  Variance-covariance or correlation matrix of a
             two-dimensional real array.
             Calculates the standard deviations and the variance-
             covariance or correlation matrix for N observations on
             each of M variables.
***LIBRARY    PMATH
***CATEGORY   L1B
***TYPE       SINGLE PRECISION (SCOVAR-S, DCOVAR-D, COVAR8-8)
***KEYWORDS   ELEMENTARY STATISTICS, STANDARD DEVIATION, VECTOR,
             VARIANCE-COVARIANCE MATRIX, CORRELATION MATRIX
***AUTHOR    Unknown, Name (LLNL/USD/NMG)
             Durst, Mark J. (LLNL/CMRD/SPG)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
             (Portable version of Cray MATHLIB routine COVARV.)
*Usage:
    INTEGER  N, M, IND
    PARAMETER (nvcv = (M*(M+1))/2)
    REAL  A(N,M), VCV(nvcv), SD(M), WK(M)
    CALL SCOVAR (A, N, M, IND, VCV, SD, WK)

*Arguments:
    A :IN      N by M array of N observations on M variables.
    N :IN      Row dimension of A.
    M :IN      Column dimension of A.
    IND:IN     Job-control flag:
                0      Return the variance-covariances.
                non-0  Return correlations.
    VCV:OUT    Array of order M(M+1)/2 containing either the
                variance-covariances or correlations in symmetric
                storage mode, depending on the value of IND.
    SD :OUT    Array of order M containing the standard deviations.
    WK:WORK    Work array of order M.

*Description:
    SCOVAR calculates the standard deviations in SD and the
    variance-covariance matrix in VCV in symmetric storage mode.  If
    IND does not equal 0, it then calls SCORRV to calculate the
    correlation matrix from the variance-covariance matrix.
    "Symmetric storage mode" means (S is taken to be the full matrix):

        VCV(k) = S(i,j), k = (1(i-1))/2 + j, i = 1,...,M, j <=1

*See Also:
    If both the variance-covariance matrix and the correlation matrix
    are required, first call SCOVAR with IND = 0. Then copy VCV into
    the desired array for the correlation matrix and call SCORRV.

***REFERENCES  (NONE)
***ROUTINES CALLED  SCORRV
***REVISION HISTORY  (YMMDD)
    830812  DATE WRITTEN
            (The above is the date of LCSD-442,Rev.1 and is undoubtedly
```

```

a significant underestimate of the age of this routine.)
890223 Added SLATEC/LDOC prologue. (FNF)
890518 Modified sequence numbers to fit in columns 73-80. (FNF)
890518 1. Replaced  $\text{expr}^{*.5}$  with  $\text{sqrt}(\text{expr})$ --one occurrence. (FNF)
      2. Corrected dimension for array VCV. (FNF)
890519 Eliminated redundant variable ink.
920319 Updated with prologue edited 891025 by G. Shaw for manual.
930706 Corrected C***CATEGORY line. (FNF)
930930 Converted old UNICOS names to S- or I-names. (DBP)
931005 Corrected list of equivalent routines, made sure that all
      variables are declared, and improved comments. (FNF)
931026 Minor changes to reduce single/double differences. (FNF)
940421 Improved purpose. (FNF)
940727 Added preprocessor directives for REAL*8 entries. (FNF)
951010 Corrected LIBRARY line. (FNF)
***END PROLOGUE  SCOVAR

```

SFITPO

```

      SUBROUTINE SFITPO (XDATA, YDATA, NDATA, NTERMS, WEIGHT, COEFF,
+                      RSD2, WORK, JOB, IERR)
***BEGIN PROLOGUE  SFITPO
***PURPOSE  Fit a polynomial to given data.
             Finds the polynomial that is the best least-squares
             fit to a given set of data points.
***LIBRARY    PMATH
***CATEGORY   K1A1A2, L8B1B1
***TYPE       SINGLE PRECISION (SFITPO-S, DFITPO-D, FITPO8-8)
***KEYWORDS   POLYNOMIAL FITTING, LEAST SQUARES
***AUTHOR    Painter, Jeffrey F., (LLNL/CMRD)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
             (Portable version of Cray MATHLIB routine FITPOL.)
*Usage:
      INTEGER  NDATA, NTERMS, JOB, IERR
      PARAMETER (NWORK = (NDATA+1)*(NTERMS+1) )
      REAL  XDATA(NDATA), YDATA(NDATA), WEIGHT(NDATA), COEFF(NTERMS),
*          RSD2, WORK(NWORK)

      CALL SFITPO (XDATA, YDATA, NDATA, NTERMS, WEIGHT, COEFF,
*                RSD2, WORK, JOB, IERR)

*Arguments:
      In the following, the data points are
      (x(i),y(i)) = ( XDATA(i), YDATA(i) ), i=1,...,NDATA.

      XDATA :IN   Array of values of the independent variable, x, among
                  which there must be at least NTERMS different values.
                  Its dimension is NDATA.

      YDATA :IN   Array of corresponding values of the dependent
                  variable, y.  Its dimension is NDATA.

      NDATA :IN   The number of data points to be fit.

      NTERMS:IN   The number of terms in the polynomial (i.e., SFITPO
                  is to determine a polynomial of degree NTERMS - 1).
                  If NTERMS > NDATA, the result will be the coefficients
                  of an interpolating polynomial of degree NDATA-1, and
                  COEFF(j) = 0 for j > NDATA.

      WEIGHT:IN   Optional weight array.

                  If WEIGHT(1) is equal to zero, SFITPO will choose
                  COEFF to minimize the sum of the squares of the
                  residuals.  In this case, WEIGHT need not be
                  dimensioned and can, indeed, be the literal 0.E0.

                  Otherwise, WEIGHT must be an array of dimension
                  NDATA, with WEIGHT(1) nonzero, and SFITPO will choose
                  COEFF to minimize the sum of the squares of the
                  weighted residuals,
                  
$$R(i) = \text{WEIGHT}(i) * (y(i) - p(x(i))), \quad i=1,2,\dots,NDATA.$$

                  (See Description, below, for definition of p(x).)

```

COEFF:OUT Array containing the NTERMS coefficients of the polynomial. COEFF(j) is the coefficient of $x^{(j-1)}$.

RSD2 :OUT Sum of the squares of the (weighted) residuals corresponding to COEFF.

WORK :WORK Array used primarily for internal computations. NWORK, its dimension, must be at least $(\text{NDATA}+1)*(\text{NTERMS}+1)$. If JOB is nonzero, the first NDATA words of WORK will contain the residuals (or weighted residuals, if the weighting option was chosen) on return:
 $\text{WORK}(i) = R(i), i = 1, 2, \dots, \text{NDATA}.$
 Note that if SREFIT is to be used for subsequent fits, WORK must not be modified in any way.

JOB :IN Residuals-computation flag:
 non-0 Residuals are computed and output in WORK.
 0 Residuals are not completely computed, although RSD2 is computed. (This option will be more efficient if the $R(i)$ are not required.)

IERR :OUT Error flag. On normal termination, IERR = 0.

Warning error: IERR <= -4
 In this case the problem looks poorly conditioned, so that all components of COEFF may be inaccurate. $10*(-\text{IERR})$ will be a lower bound for the condition number, and COEFF will be computed anyway. (See "Accuracy" below for details.)

Fatal error:
 SQRSL returned INFO=IERR: $0 < \text{IERR} \leq \text{NTERMS}$
 A singular matrix has been detected. This may be due to too many values of XDATA(i) exactly equal or too many weights equal to zero.
 COEFF has not been computed in this case.

***Description:**

SFITPO finds the polynomial that is the best least-squares fit to a given set of data points

$$(x(i), y(i)) = (XDATA(i), YDATA(i)), \quad i = 1, 2, \dots, \text{NDATA}.$$

It finds coefficients COEFF(1), ..., COEFF(NTERMS) of the polynomial

$$y = p(x) = \text{COEFF}(1) + \text{COEFF}(2)*x + \text{COEFF}(3)*x^2 + \dots + \text{COEFF}(\text{NTERMS})*x^{(\text{NTERMS}-1)},$$

which minimize the sum of the squares of the residuals

$$R(i) = y(i) - p(x(i)), \quad i = 1, 2, \dots, \text{NDATA}.$$

As an option, the residuals may be weighted, as noted above.

If the range of x-values is far from zero, SFITPO may introduce extra inaccuracies in the results, especially in lower-order coefficients. A way to get better results is to choose a typical value of x, say x_0 , and define

```
xnew(i) = x(i) - x0,  i = 1, 2, ..., NDATA .
```

Then instead of

```
CALL SFITPO (x, ...)
```

use

```
CALL SFITPO (xnew, ...) .
```

The result will be coefficients for the polynomial

```
y = p(xnew) = p(x-x0) .
```

Let A denote the matrix whose i-th row is

```
( 1  XDATA(i)  XDATA(i)**2  ....  XDATA(i)**(NTERMS-1) )
```

(This row is multiplied by WEIGHT(i) if the weighting option has been chosen.) A is called the least-squares matrix. The solution to the least-squares problem is found by way of a QR decomposition of A, without pivoting, using LINPACK routines SQRDC and SQRSL.

The covariance matrix of COEFF can be estimated after a call of SFITPO. If all the data points $y = YDATA(i)$ have the same variance $v(y)$, then the covariance matrix is $v(y)$ times the inverse of the product of A-transpose (denoted A^t) and A:

```
cov = v(y) * inv(At*A) ,
```

An estimate of $v(y)$ is $RSD2/(NDATA - NTERMS)$. The following call of a LINPACK subroutine (Ref. 1) will compute $inv(A^t A)$:

```
CALL SPODI (WORK(2+NDATA), NDATA, NTERMS, DUMMY, 1)
```

where WORK, NDATA, and NTERMS are the same variables as in SFITPO, WORK has not been disturbed since the last SFITPO call, and DUMMY is not referenced. Only WORK is changed. For $i \leq j$, SPODI puts the (i,j)th element of $inv(A^t A)$ (which equals the (j,i)th element) into $WORK(i+j*NDATA+1)$. CAUTION: Since this changes WORK, SREFIT cannot be called after such a call of SPODI.

Sometimes an expression involving $inv(A^t A)$ can be evaluated without computing the inverse; if so, and if NTERMS is large, it will be cheaper not to compute the inverse. An equation of the form

```
(At*A) * w = b
```

can best be solved for w by the following call of a LINPACK routine (Ref. 1):

```
CALL SPOSL (WORK(2+NDATA), NDATA, NTERMS, BW)
```

where WORK, NDATA, and NTERMS are input variables, undisturbed since the last SFITPO call, and BW is a real vector of dimension NTERMS. On input, BW is b, and on output, it is w. Since SPOSL does not change WORK, you may call SREFIT or SPOSL after calling

SPOSL.

***Examples:**

See the SREFIT writeup for a sample call of SFITPO.

The following sample code is a faster way to evaluate the polynomial $Y = p(X)$ than the most straightforward approach.

```
      Y = COEFF(NTERMS)
      DO 10 J = 1, (NTERMS - 1)
10    Y = X*Y + COEFF(NTERMS - J)
```

***Accuracy:**

SFITPO finds a lower bound for the condition number K of the problem. This number is relevant because SFITPO will introduce an error in each $\text{COEFF}(j)$ ($j = 1, 2, \dots, \text{NTERMS}$) that is roughly proportional to K times the largest of these coefficients (larger if there are large values of x in the data). If the condition-number estimate is over 10,000, then the error flag IERR will be set to a negative number so that K is greater than $10^{**}(|\text{IERR}|)$. It is unlikely that K will be any larger than $10^{**}(|\text{IERR}| + 2)$. As a rule of thumb, this means that the largest of the coefficients may have lost about $|\text{IERR}| + 2$ digits of accuracy. The same absolute error estimate applies to all of the coefficients; thus, if $\text{COEFF}(j)$ is smaller than the largest coefficient by a factor of $10^{**}n$, it will have lost $|\text{IERR}| + 2 + n$ digits of accuracy. If some values of x are large and if NTERMS is large, then lower-order coefficients will be less accurate. For details, see Ref. 1, pp. I.8-I.11 and 9.4-9.5, and Ref. 2, pp. 28-35.

The above discussion applies to the mathematical fitting problem; of course there may be other inaccuracies from the input data. Furthermore, the polynomial computed when $\text{IERR} < 0$ may be perfectly acceptable if all one needs is a function that produces small residuals.

***Cautions:**

SFITPO assumes $1 \leq \text{NTERMS}$, NDATA . This is not checked. See description of NTERMS for behavior when $\text{NTERMS} > \text{NDATA}$.

This is a simple program for simple problems. It is not recommended for large problems.

***Portability:**

This routine calls the LINPACK routines SQRDC and SQRSL , and BLAS (Basic Linear Algebra Subprograms) SDOT .

The declaration $\text{REAL WORK}(\text{NDATA},*)$ is used to cause the compiler to generate suitable subscript arithmetic for the NDATA by NTERMS least-squares matrix stored starting at element $\text{WORK}(2,2) = \text{WORK}(\text{NDATA}+2)$. Some compilers may object to the fact that $(\text{I}+1) > \text{NDATA}$ when $\text{I} = \text{NDATA}$ in loops 10, 30 and 50.

***ROUTINES CALLED SDOT, SQRDC, SQRSL

***REVISION HISTORY (YYMMDD)

800301 DATE WRITTEN

890419 Added SLATEC/LDOC prologue. (FNF)

890424 Corrected DATE WRITTEN. (FNF)

890518 Modified sequence numbers to fit in columns 73-80. (FNF)

920319 Updated with prologue edited 891025 by G. Shaw for manual.
920331 Reformatted references section. (FNF)
930706 Corrected C***CATEGORY line. (FNF)
930930 Converted old UNICOS names to S- or I-names. (DBP)
931005 Augmented list of equivalent routines, made sure that all
variables are declared, and improved comments. (FNF)
931026 Minor changes to reduce single/double differences. (FNF)
951106 Added special treatment for NTERMS > NDATA. (FNF)
***END PROLOGUE SFITPO

SLSODE

```
      SUBROUTINE SLSODE (F, NEQ, Y, T, TOUT, ITOL, RTOL, ATOL, ITASK,  
+                      ISTATE, IOPT, RWORK, LRW, IWORK, LIW, JAC, MF)  
***BEGIN PROLOGUE  SLSODE  
***PURPOSE  Livermore solver for ordinary differential equations.  
            Solves the initial-value problem for stiff or nonstiff  
            systems of first-order ODE's,  
            dy/dt = f(t,y),    or, in component form,  
            dy(i)/dt = f(i) = f(i,t,y(1),y(2),...,y(N)),  i=1,...,N.  
***LIBRARY  PMATH (ODEPACK)  
***CATEGORY  I1A1B, I1A2  
***TYPE      SINGLE PRECISION (SLSODE-S, DLSODE-D, LSODE8-8)  
***KEYWORDS  ORDINARY DIFFERENTIAL EQUATIONS, INITIAL VALUE PROBLEM,  
            STIFF, NONSTIFF  
***AUTHOR  Hindmarsh, Alan C., (LLNL)  
            Center for Computational Sciences and Engrg., L-316  
            Lawrence Livermore National Laboratory  
            Livermore, CA 94550.  
***DESCRIPTION  
            (Portable version of Cray MATHLIB routine LSODE.)
```

NOTE: The SLSODE solver is not re-entrant, and so is usable on the Cray multi-processor machines only if it is not used in a multi-tasking environment. If re-entrancy is required, use NLSODE instead.

The formats of the SLSODE and NLSODE writeups differ from those of the other MATHLIB routines.

The "Usage" and "Arguments" sections treat only a subset of available options, in condensed fashion. The options covered and the information supplied will support most standard uses of SLSODE.

For more sophisticated uses, full details on all options are given in the concluding section, headed "Long Description." A synopsis of the SLSODE Long Description is provided at the beginning of that section; general topics covered are:

- Elements of the call sequence; optional input and output
- Optional supplemental routines in the SLSODE package
- internal COMMON block

*Usage:

Communication between the user and the SLSODE package, for normal situations, is summarized here. This summary describes a subset of the available options. See "Long Description" for complete details, including optional communication, nonstandard options, and instructions for special situations.

A sample program is given in the "Examples" section.

Refer to the argument descriptions for the definitions of the quantities that appear in the following sample declarations.

```
For MF = 10,  
    PARAMETER (LRW = 20 + 16*NEQ,          LIW = 20)  
For MF = 21 or 22,
```

```

      PARAMETER (LRW = 22 + 9*NEQ + NEQ**2, LIW = 20 + NEQ)
For MF = 24 or 25,
      PARAMETER (LRW = 22 + 10*NEQ + (2*ML+MU)*NEQ,
*
*                                     LIW = 20 + NEQ)

      EXTERNAL F, JAC
      INTEGER NEQ, ITOL, ITASK, ISTATE, IOPT, LRW, IWORK(LIW),
*
*      LIW, MF
      REAL Y(NEQ), T, TOUT, RTOL, ATOL(NTOL), RWORK(LRW)

      CALL SLSODE (F, NEQ, Y, T, TOUT, ITOL, RTOL, ATOL, ITASK,
*
*      ISTATE, IOPT, RWORK, LRW, IWORK, LIW, JAC, MF)

```

***Arguments:**

F :EXT Name of subroutine for right-hand-side vector f. This name must be declared EXTERNAL in calling program. The form of F must be:

```

      SUBROUTINE F (NEQ, T, Y, YDOT)
      INTEGER NEQ
      REAL T, Y(NEQ), YDOT(NEQ)

```

The inputs are NEQ, T, Y. F is to set

$$YDOT(i) = f(i, T, Y(1), Y(2), \dots, Y(NEQ)),$$

i = 1, ..., NEQ .

NEQ :IN Number of first-order ODE's.

Y :INOUT Array of values of the y(t) vector, of length NEQ.
 Input: For the first call, Y should contain the values of y(t) at t = T. (Y is an input variable only if ISTATE = 1.)
 Output: On return, Y will contain the values at the new t-value.

T :INOUT Value of the independent variable. On return it will be the current value of t (normally TOUT).

TOUT :IN Next point where output is desired (.NE. T).

ITOL :IN 1 or 2 according as ATOL (below) is a scalar or an array.

RTOL :IN Relative tolerance parameter (scalar).

ATOL :IN Absolute tolerance parameter (scalar or array).
 If ITOL = 1, ATOL need not be dimensioned.
 If ITOL = 2, ATOL must be dimensioned at least NEQ.

The estimated local error in Y(i) will be controlled so as to be roughly less (in magnitude) than

$$\begin{aligned} EWT(i) &= RTOL * ABS(Y(i)) + ATOL && \text{if ITOL} = 1, \text{ or} \\ EWT(i) &= RTOL * ABS(Y(i)) + ATOL(i) && \text{if ITOL} = 2. \end{aligned}$$

Thus the local error test passes if, in each component, either the absolute error is less than ATOL (or ATOL(i)), or the relative error is less than RTOL.

Use $RTOL = 0.0$ for pure absolute error control, and use $ATOL = 0.0$ (or $ATOL(i) = 0.0$) for pure relative error control. Caution: Actual (global) errors may exceed these local tolerances, so choose them conservatively.

ITASK :IN Flag indicating the task SLSODE is to perform. Use ITASK = 1 for normal computation of output values of y at t = TOUT.

ISTATE:INOUT Index used for input and output to specify the state of the calculation.

Input:

- 1 This is the first call for a problem.
- 2 This is a subsequent call.

Output:

- 2 SLSODE was successful (otherwise, negative). Note that ISTATE need not be modified after a successful return.
- 1 Excess work done on this call (perhaps wrong MF).
- 2 Excess accuracy requested (tolerances too small).
- 3 Illegal input detected (see printed message).
- 4 Repeated error test failures (check all inputs).
- 5 Repeated convergence failures (perhaps bad Jacobian supplied or wrong choice of MF or tolerances).
- 6 Error weight became zero during problem (solution component i vanished, and ATOL or $ATOL(i) = 0.$).

IOPT :IN Flag indicating whether optional inputs are used:

- 0 No.
- 1 Yes. (See "Optional inputs" under "Long Description," Part 1.)

RWORK :WORK Real work array of length at least:

- 20 + 16*NEQ for MF = 10,
- 22 + 9*NEQ + NEQ**2 for MF = 21 or 22,
- 22 + 10*NEQ + (2*ML + MU)*NEQ for MF = 24 or 25.

LRW :IN Declared length of RWORK (in user's DIMENSION statement).

IWORK :WORK Integer work array of length at least:

- 20 for MF = 10,
- 20 + NEQ for MF = 21, 22, 24, or 25.

If MF = 24 or 25, input in IWORK(1), IWORK(2) the lower and upper Jacobian half-bandwidths ML, MU.

On return, IWORK contains information that may be of interest to the user:

Name	Location	Meaning
NST	IWORK(11)	Number of steps taken for the problem so

		far.
NFE	IWORK(12)	Number of f evaluations for the problem so far.
NJE	IWORK(13)	Number of Jacobian evaluations (and of matrix LU decompositions) for the problem so far.
NQU	IWORK(14)	Method order last used (successfully).
LENRW	IWORK(17)	Length of RWORK actually required. This is defined on normal returns and on an illegal input return for insufficient storage.
LENIW	IWORK(18)	Length of IWORK actually required. This is defined on normal returns and on an illegal input return for insufficient storage.
LIW	:IN	Declared length of IWORK (in user's DIMENSION statement).
JAC	:EXT	Name of subroutine for Jacobian matrix (MF = 21 or 24). If used, this name must be declared EXTERNAL in calling program. If not used, pass a dummy name. The form of JAC must be: <pre> SUBROUTINE JAC (NEQ, T, Y, ML, MU, PD, NROWPD) INTEGER NEQ, ML, MU, NROWPD REAL T, Y(NEQ), PD(NROWPD,NEQ) </pre> See item c, under "Description" below for more information about JAC.
MF	:IN	Method flag. Standard values are: 10 Nonstiff (Adams) method, no Jacobian used. 21 Stiff (BDF) method, user-supplied full Jacobian. 22 Stiff method, internally generated full Jacobian. 24 Stiff method, user-supplied banded Jacobian. 25 Stiff method, internally generated banded Jacobian.

***Description:**

SLSODE solves the initial value problem for stiff or nonstiff systems of first-order ODE's,

$$dy/dt = f(t,y) ,$$

or, in component form,

$$dy(i)/dt = f(i) = f(i,t,y(1),y(2),\dots,y(NEQ)) \\ (i = 1, \dots, NEQ) .$$

SLSODE is a package based on the GEAR and GEARB packages, and on the October 23, 1978, version of the tentative ODEPACK user interface standard, with minor modifications.

The steps in solving such a problem are as follows.

a. First write a subroutine of the form

```

SUBROUTINE F (NEQ, T, Y, YDOT)

```

PMATH Reference Manual - 109

```

      INTEGER  NEQ
      REAL    T, Y(NEQ), YDOT(NEQ)

```

which supplies the vector function f by loading $YDOT(i)$ with $f(i)$.

- b. Next determine (or guess) whether or not the problem is stiff. Stiffness occurs when the Jacobian matrix df/dy has an eigenvalue whose real part is negative and large in magnitude compared to the reciprocal of the t span of interest. If the problem is nonstiff, use method flag $MF = 10$. If it is stiff, there are four standard choices for MF , and SLSODE requires the Jacobian matrix in some form. This matrix is regarded either as full ($MF = 21$ or 22), or banded ($MF = 24$ or 25). In the banded case, SLSODE requires two half-bandwidth parameters ML and MU . These are, respectively, the widths of the lower and upper parts of the band, excluding the main diagonal. Thus the band consists of the locations (i,j) with

$$i - ML \leq j \leq i + MU ,$$

and the full bandwidth is $ML + MU + 1$.

- c. If the problem is stiff, you are encouraged to supply the Jacobian directly ($MF = 21$ or 24), but if this is not feasible, SLSODE will compute it internally by difference quotients ($MF = 22$ or 25). If you are supplying the Jacobian, write a subroutine of the form

```

      SUBROUTINE JAC (NEQ, T, Y, ML, MU, PD, NROWPD)
      INTEGER  NEQ, ML, MU, NRWOPD
      REAL    Y, Y(NEQ), PD(NROWPD,NEQ)

```

which provides df/dy by loading PD as follows:

- For a full Jacobian ($MF = 21$), load $PD(i,j)$ with $df(i)/dy(j)$, the partial derivative of $f(i)$ with respect to $y(j)$. (Ignore the ML and MU arguments in this case.)
 - For a banded Jacobian ($MF = 24$), load $PD(i-j+MU+1,j)$ with $df(i)/dy(j)$; i.e., load the diagonal lines of df/dy into the rows of PD from the top down.
 - In either case, only nonzero elements need be loaded.
- d. Write a main program that calls subroutine SLSODE once for each point at which answers are desired. This should also provide for possible use of logical unit 6 for output of error messages by SLSODE.

Before the first call to SLSODE, set $ISTATE = 1$, set Y and T to the initial values, and set $TOUT$ to the first output point. To continue the integration after a successful return, simply reset $TOUT$ and call SLSODE again. No other parameters need be reset.

*Examples:

The following is a simple example problem, with the coding needed for its solution by SLSODE. The problem is from chemical kinetics, and consists of the following three rate equations:

$$\begin{aligned}
 dy_1/dt &= -.04*y_1 + 1.E4*y_2*y_3 \\
 dy_2/dt &= .04*y_1 - 1.E4*y_2*y_3 - 3.E7*y_2**2
 \end{aligned}$$

$$dy_3/dt = 3.E7*y_2**2$$

on the interval from $t = 0.0$ to $t = 4.E10$, with initial conditions $y_1 = 1.0$, $y_2 = y_3 = 0$. The problem is stiff.

The following coding solves this problem with SLSODE, using $MF = 21$ and printing results at $t = .4, 4., \dots, 4.E10$. It uses $ITOL = 2$ and $ATOL$ much smaller for y_2 than for y_1 or y_3 because y_2 has much smaller values. At the end of the run, statistical quantities of interest are printed.

```

      EXTERNAL  FEX, JEX
      INTEGER  IOPT, IOUT, ISTATE, ITASK, ITOL, IWORK(23), LIW, LRW,
*            MF, NEQ
      REAL  ATOL(3), RTOL, RWORK(58), T, TOUT, Y(3)
      NEQ = 3
      Y(1) = 1.
      Y(2) = 0.
      Y(3) = 0.
      T = 0.
      TOUT = .4
      ITOL = 2
      RTOL = 1.E-4
      ATOL(1) = 1.E-6
      ATOL(2) = 1.E-10
      ATOL(3) = 1.E-6
      ITASK = 1
      ISTATE = 1
      IOPT = 0
      LRW = 58
      LIW = 23
      MF = 21
      DO 40 IOUT = 1,12
        CALL SLSODE (FEX, NEQ, Y, T, TOUT, ITOL, RTOL, ATOL, ITASK,
*                  ISTATE, IOPT, RWORK, LRW, IWORK, LIW, JEX, MF)
        WRITE(6,20)  T, Y(1), Y(2), Y(3)
20      FORMAT(' At t =',E12.4,'    y =',3E14.6)
        IF (ISTATE .LT. 0)  GO TO 80
40      TOUT = TOUT*10.
        WRITE(6,60)  IWORK(11), IWORK(12), IWORK(13)
60      FORMAT('/' No. steps =',i4,',  No. f-s =',i4,',  No. J-s =',i4)
        STOP
80      WRITE(6,90)  ISTATE
90      FORMAT('Error halt.. ISTATE =',I3)
        STOP
      END

      SUBROUTINE  FEX (NEQ, T, Y, YDOT)
      INTEGER  NEQ
      REAL  T, Y(3), YDOT(3)
      YDOT(1) = -.04*Y(1) + 1.E4*Y(2)*Y(3)
      YDOT(3) = 3.E7*Y(2)*Y(2)
      YDOT(2) = -YDOT(1) - YDOT(3)
      RETURN
      END

      SUBROUTINE  JEX (NEQ, T, Y, ML, MU, PD, NRPD)
      INTEGER  NEQ, ML, MU, NRPD
      REAL  T, Y(3), PD(NRPD,3)
      PD(1,1) = -.04

```

```

PD(1,2) = 1.E4*Y(3)
PD(1,3) = 1.E4*Y(2)
PD(2,1) = .04
PD(2,3) = -PD(1,3)
PD(3,2) = 6.E7*Y(2)
PD(2,2) = -PD(1,2) - PD(3,2)
RETURN
END

```

The output from this program (on a Cray-1 in single precision) is as follows.

At t =	4.0000e-01	y =	9.851726e-01	3.386406e-05	1.479357e-02
At t =	4.0000e+00	y =	9.055142e-01	2.240418e-05	9.446344e-02
At t =	4.0000e+01	y =	7.158050e-01	9.184616e-06	2.841858e-01
At t =	4.0000e+02	y =	4.504846e-01	3.222434e-06	5.495122e-01
At t =	4.0000e+03	y =	1.831701e-01	8.940379e-07	8.168290e-01
At t =	4.0000e+04	y =	3.897016e-02	1.621193e-07	9.610297e-01
At t =	4.0000e+05	y =	4.935213e-03	1.983756e-08	9.950648e-01
At t =	4.0000e+06	y =	5.159269e-04	2.064759e-09	9.994841e-01
At t =	4.0000e+07	y =	5.306413e-05	2.122677e-10	9.999469e-01
At t =	4.0000e+08	y =	5.494530e-06	2.197825e-11	9.999945e-01
At t =	4.0000e+09	y =	5.129458e-07	2.051784e-12	9.999995e-01
At t =	4.0000e+10	y =	-7.170603e-08	-2.868241e-13	1.000000e+00

No. steps = 330, No. f-s = 405, No. J-s = 69

***Accuracy:**

The accuracy of the solution depends on the choice of tolerances RTOL and ATOL. Actual (global) errors may exceed these local tolerances, so choose them conservatively.

***Cautions:**

The work arrays should not be altered between calls to SLSODE for the same problem, except possibly for the conditional and optional inputs.

***Portability:**

Since NEQ is dimensioned inside SLSODE, some compilers may object to a call to SLSODE with NEQ a scalar variable. In this event, use DIMENSION NEQ(1). Similar remarks apply to RTOL and ATOL.

Note to Cray users:

For maximum efficiency, use the CFT77 compiler. Appropriate compiler optimization directives have been inserted for CFT77 (but not CIVIC).

NOTICE: If moving the SLSODE source code to other systems, contact the author for notes on nonstandard Fortran usage, COMMON block, and other installation details.

***Reference:**

Alan C. Hindmarsh, "ODEPACK, a systematized collection of ODE solvers," in Scientific Computing, R. S. Stepleman, et al., Eds. (North-Holland, Amsterdam, 1983), pp. 55-64.

***Long Description:**

The following complete description of the user interface to SLSODE consists of four parts:

1. The call sequence to subroutine SLSODE, which is a driver routine for the solver. This includes descriptions of both the call sequence arguments and user-supplied routines. Following these descriptions is a description of optional inputs available through the call sequence, and then a description of optional outputs in the work arrays.
2. Descriptions of other routines in the SLSODE package that may be (optionally) called by the user. These provide the ability to alter error message handling, save and restore the internal COMMON, and obtain specified derivatives of the solution $y(t)$.
3. Descriptions of COMMON block to be declared in overlay or similar environments, or to be saved when doing an interrupt of the problem and continued solution later.
4. Description of two routines in the SLSODE package, either of which the user may replace with his own version, if desired. These relate to the measurement of errors.

Part 1. Call Sequence

Arguments

The call sequence parameters used for input only are

F, NEQ, TOUT, ITOL, RTOL, ATOL, ITASK, IOPT, LRW, LIW, JAC, MF,

and those used for both input and output are

Y, T, ISTATE.

The work arrays RWORK and IWORK are also used for conditional and optional inputs and optional outputs. (The term output here refers to the return from subroutine SLSODE to the user's calling program.)

The legality of input parameters will be thoroughly checked on the initial call for the problem, but not checked thereafter unless a change in input parameters is flagged by ISTATE = 3 on input.

The descriptions of the call arguments are as follows.

F The name of the user-supplied subroutine defining the ODE system. The system must be put in the first-order form $dy/dt = f(t,y)$, where f is a vector-valued function of the scalar t and the vector y . Subroutine F is to compute the function f . It is to have the form

```
SUBROUTINE F (NEQ, T, Y, YDOT)
  REAL Y(NEQ), YDOT(NEQ)
```

where NEQ, T, and Y are input, and the array YDOT = $f(T,Y)$ is output. Y and YDOT are arrays of length NEQ. Subroutine F should not alter $Y(1), \dots, Y(NEQ)$. F must be declared EXTERNAL in the calling program.

Subroutine F may access user-defined quantities in

NEQ(2),... and/or in Y(NEQ(1)+1),..., if NEQ is an array (dimensioned in F) and/or Y has length exceeding NEQ(1). See the descriptions of NEQ and Y below.

If quantities computed in the F routine are needed externally to SLSODE, an extra call to F should be made for this purpose, for consistent and accurate results. If only the derivative dy/dt is needed, use SINTDY instead.

NEQ The size of the ODE system (number of first-order ordinary differential equations). Used only for input. NEQ may be decreased, but not increased, during the problem. If NEQ is decreased (with ISTATE = 3 on input), the remaining components of Y should be left undisturbed, if these are to be accessed in F and/or JAC.

Normally, NEQ is a scalar, and it is generally referred to as a scalar in this user interface description. However, NEQ may be an array, with NEQ(1) set to the system size. (The SLSODE package accesses only NEQ(1).) In either case, this parameter is passed as the NEQ argument in all calls to F and JAC. Hence, if it is an array, locations NEQ(2),... may be used to store other integer data and pass it to F and/or JAC. Subroutines F and/or JAC must include NEQ in a DIMENSION statement in that case.

Y A real array for the vector of dependent variables, of length NEQ or more. Used for both input and output on the first call (ISTATE = 1), and only for output on other calls. On the first call, Y must contain the vector of initial values. On output, Y contains the computed solution vector, evaluated at T. If desired, the Y array may be used for other purposes between calls to the solver.

This array is passed as the Y argument in all calls to F and JAC. Hence its length may exceed NEQ, and locations Y(NEQ+1),... may be used to store other real data and pass it to F and/or JAC. (The SLSODE package accesses only Y(1),...,Y(NEQ).)

T The independent variable. On input, T is used only on the first call, as the initial point of the integration. On output, after each call, T is the value at which a computed solution Y is evaluated (usually the same as TOUT). On an error return, T is the farthest point reached.

TOUT The next value of T at which a computed solution is desired. Used only for input.

When starting the problem (ISTATE = 1), TOUT may be equal to T for one call, then should not equal T for the next call. For the initial T, an input value of TOUT .NE. T is used in order to determine the direction of the integration (i.e., the algebraic sign of the step sizes) and the rough scale of the problem. Integration in either direction (forward or backward in T) is permitted.

If ITASK = 2 or 5 (one-step modes), TOUT is ignored after the first call (i.e., the first call with TOUT .NE. T). Otherwise, TOUT is required on every call.

If ITASK = 1, 3, or 4, the values of TOUT need not be monotone, but a value of TOUT which backs up is limited to the current internal T interval, whose endpoints are TCUR - HU and TCUR. (See "Optional Outputs" below for TCUR and HU.)

- ITOL An indicator for the type of error control. See description below under ATOL. Used only for input.
- RTOL A relative error tolerance parameter, either a scalar or an array of length NEQ. See description below under ATOL. Input only.
- ATOL An absolute error tolerance parameter, either a scalar or an array of length NEQ. Input only.

The input parameters ITOL, RTOL, and ATOL determine the error control performed by the solver. The solver will control the vector $e = (e(i))$ of estimated local errors in Y, according to an inequality of the form

$$\text{rms-norm of } (e(i)/EWT(i)) \leq 1,$$

where

$$EWT(i) = RTOL(i)*ABS(Y(i)) + ATOL(i),$$

and the rms-norm (root-mean-square norm) here is

$$\text{rms-norm}(v) = \text{SQRT}(\text{sum } v(i)**2 / \text{NEQ}).$$

Here $EWT = (EWT(i))$ is a vector of weights which must always be positive, and the values of RTOL and ATOL should all be nonnegative. The following table gives the types (scalar/array) of RTOL and ATOL, and the corresponding form of $EWT(i)$.

ITOL	RTOL	ATOL	EWT(i)
----	-----	-----	-----
1	scalar	scalar	$RTOL*ABS(Y(i)) + ATOL$
2	scalar	array	$RTOL*ABS(Y(i)) + ATOL(i)$
3	array	scalar	$RTOL(i)*ABS(Y(i)) + ATOL$
4	array	array	$RTOL(i)*ABS(Y(i)) + ATOL(i)$

When either of these parameters is a scalar, it need not be dimensioned in the user's calling program.

If none of the above choices (with ITOL, RTOL, and ATOL fixed throughout the problem) is suitable, more general error controls can be obtained by substituting user-supplied routines for the setting of EWT and/or for the norm calculation. See Part 4 below.

If global errors are to be estimated by making a repeated

run on the same problem with smaller tolerances, then all components of RTOL and ATOL (i.e., of EWT) should be scaled down uniformly.

ITASK An index specifying the task to be performed. Input only. ITASK has the following values and meanings:

- 1 Normal computation of output values of $y(t)$ at $t = TOUT$ (by overshooting and interpolating).
- 2 Take one step only and return.
- 3 Stop at the first internal mesh point at or beyond $t = TOUT$ and return.
- 4 Normal computation of output values of $y(t)$ at $t = TOUT$ but without overshooting $t = TCRIT$. TCRIT must be input as RWORK(1). TCRIT may be equal to or beyond TOUT, but not behind it in the direction of integration. This option is useful if the problem has a singularity at or beyond $t = TCRIT$.
- 5 Take one step, without passing TCRIT, and return. TCRIT must be input as RWORK(1).

Note: If ITASK = 4 or 5 and the solver reaches TCRIT (within roundoff), it will return $T = TCRIT$ (exactly) to indicate this (unless ITASK = 4 and TOUT comes before TCRIT, in which case answers at $T = TOUT$ are returned first).

ISTATE An index used for input and output to specify the state of the calculation.

On input, the values of ISTATE are as follows:

- 1 This is the first call for the problem (initializations will be done). See "Note" below.
- 2 This is not the first call, and the calculation is to continue normally, with no change in any input parameters except possibly TOUT and ITASK. (If ITOL, RTOL, and/or ATOL are changed between calls with ISTATE = 2, the new values will be used but not tested for legality.)
- 3 This is not the first call, and the calculation is to continue normally, but with a change in input parameters other than TOUT and ITASK. Changes are allowed in NEQ, ITOL, RTOL, ATOL, IOPT, LRW, LIW, MF, ML, MU, and any of the optional inputs except H0. (See IWORK description for ML and MU.)

Note: A preliminary call with $TOUT = T$ is not counted as a first call here, as no initialization or checking of input is done. (Such a call is sometimes useful for the purpose of outputting the initial conditions.) Thus the first call for which $TOUT \neq T$ requires ISTATE = 1 on input.

On output, ISTATE has the following values and meanings:

- 1 Nothing was done, as TOUT was equal to T with ISTATE = 1 on input.
- 2 The integration was performed successfully.
- 1 An excessive amount of work (more than MXSTEP steps) was done on this call, before completing the requested task, but the integration was otherwise successful as far as T. (MXSTEP is an optional input

and is normally 500.) To continue, the user may simply reset ISTATE to a value >1 and call again (the excess work step counter will be reset to 0). In addition, the user may increase MXSTEP to avoid this error return; see "Optional Inputs" below.

- 2 Too much accuracy was requested for the precision of the machine being used. This was detected before completing the requested task, but the integration was successful as far as T. To continue, the tolerance parameters must be reset, and ISTATE must be set to 3. The optional output TOLSF may be used for this purpose. (Note: If this condition is detected before taking any steps, then an illegal input return (ISTATE = -3) occurs instead.)
- 3 Illegal input was detected, before taking any integration steps. See written message for details. (Note: If the solver detects an infinite loop of calls to the solver with illegal input, it will cause the run to stop.)
- 4 There were repeated error-test failures on one attempted step, before completing the requested task, but the integration was successful as far as T. The problem may have a singularity, or the input may be inappropriate.
- 5 There were repeated convergence-test failures on one attempted step, before completing the requested task, but the integration was successful as far as T. This may be caused by an inaccurate Jacobian matrix, if one is being used.
- 6 EWT(i) became zero for some i during the integration. Pure relative error control (ATOL(i)=0.0) was requested on a variable which has now vanished. The integration was successful as far as T.

Note: Since the normal output value of ISTATE is 2, it does not need to be reset for normal continuation. Also, since a negative input value of ISTATE will be regarded as illegal, a negative output value requires the user to change it, and possibly other inputs, before calling the solver again.

IOPT An integer flag to specify whether any optional inputs are being used on this call. Input only. The optional inputs are listed under a separate heading below.
 0 No optional inputs are being used. Default values will be used in all cases.
 1 One or more optional inputs are being used.

RWORK A real working array (single precision). The length of RWORK must be at least

$$20 + NYH*(MAXORD + 1) + 3*NEQ + LWM$$

where

NYH = the initial value of NEQ,
 MAXORD = 12 (if METH = 1) or 5 (if METH = 2) (unless a smaller value is given as an optional input),
 LWM = 0 if MITER = 0,
 LWM = NEQ**2 + 2 if MITER = 1 or 2,
 LWM = NEQ + 2 if MITER = 3, and

$$\text{LWM} = (2 * \text{ML} + \text{MU} + 1) * \text{NEQ} + 2$$

if MITER = 4 or 5.

(See the MF description below for METH and MITER.)

Thus if MAXORD has its default value and NEQ is constant, this length is:

20 + 16*NEQ	for MF = 10,
22 + 16*NEQ + NEQ**2	for MF = 11 or 12,
22 + 17*NEQ	for MF = 13,
22 + 17*NEQ + (2*ML + MU)*NEQ	for MF = 14 or 15,
20 + 9*NEQ	for MF = 20,
22 + 9*NEQ + NEQ**2	for MF = 21 or 22,
22 + 10*NEQ	for MF = 23,
22 + 10*NEQ + (2*ML + MU)*NEQ	for MF = 24 or 25.

The first 20 words of RWORK are reserved for conditional and optional inputs and optional outputs.

The following word in RWORK is a conditional input:
 RWORK(1) = TCRIT, the critical value of t which the solver is not to overshoot. Required if ITASK is 4 or 5, and ignored otherwise. See ITASK.

LRW The length of the array RWORK, as declared by the user.
 (This will be checked by the solver.)

IWORK An integer work array. Its length must be at least
 20 if MITER = 0 or 3 (MF = 10, 13, 20, 23), or
 20 + NEQ otherwise (MF = 11, 12, 14, 15, 21, 22, 24, 25).
 (See the MF description below for MITER.) The first few
 words of IWORK are used for conditional and optional
 inputs and optional outputs.

The following two words in IWORK are conditional inputs:
 IWORK(1) = ML These are the lower and upper half-
 IWORK(2) = MU bandwidths, respectively, of the banded
 Jacobian, excluding the main diagonal.
 The band is defined by the matrix locations
 (i,j) with $i - \text{ML} \leq j \leq i + \text{MU}$. ML and MU
 must satisfy $0 \leq \text{ML}, \text{MU} \leq \text{NEQ} - 1$. These are
 required if MITER is 4 or 5, and ignored
 otherwise. ML and MU may in fact be the band
 parameters for a matrix to which df/dy is only
 approximately equal.

LIW The length of the array IWORK, as declared by the user.
 (This will be checked by the solver.)

Note: The work arrays must not be altered between calls to SLSODE for the same problem, except possibly for the conditional and optional inputs, and except for the last 3*NEQ words of RWORK. The latter space is used for internal scratch space, and so is available for use by the user outside SLSODE between calls, if desired (but not for use by F or JAC).

JAC The name of the user-supplied routine (MITER = 1 or 4) to compute the Jacobian matrix, df/dy , as a function of the scalar t and the vector y. (See the MF description below for MITER.) It is to have the form

```

SUBROUTINE JAC (NEQ, T, Y, ML, MU, PD, NROWPD)
REAL Y(NEQ), PD(NROWPD,NEQ)

```

where NEQ, T, Y, ML, MU, and NROWPD are input and the array PD is to be loaded with partial derivatives (elements of the Jacobian matrix) on output. PD must be given a first dimension of NROWPD. T and Y have the same meaning as in subroutine F.

In the full matrix case (MITER = 1), ML and MU are ignored, and the Jacobian is to be loaded into PD in columnwise manner, with $df(i)/dy(j)$ loaded into PD(i,j).

In the band matrix case (MITER = 4), the elements within the band are to be loaded into PD in columnwise manner, with diagonal lines of df/dy loaded into the rows of PD. Thus $df(i)/dy(j)$ is to be loaded into PD(i-j+MU+1,j). ML and MU are the half-bandwidth parameters (see IWORK). The locations in PD in the two triangular areas which correspond to nonexistent matrix elements can be ignored or loaded arbitrarily, as they are overwritten by SLSODE.

JAC need not provide df/dy exactly. A crude approximation (possibly with a smaller bandwidth) will do.

In either case, PD is preset to zero by the solver, so that only the nonzero elements need be loaded by JAC. Each call to JAC is preceded by a call to F with the same arguments NEQ, T, and Y. Thus to gain some efficiency, intermediate quantities shared by both calculations may be saved in a user COMMON block by F and not recomputed by JAC, if desired. Also, JAC may alter the Y array, if desired. JAC must be declared EXTERNAL in the calling program.

Subroutine JAC may access user-defined quantities in NEQ(2),... and/or in Y(NEQ(1)+1),... if NEQ is an array (dimensioned in JAC) and/or Y has length exceeding NEQ(1). See the descriptions of NEQ and Y above.

MF The method flag. Used only for input. The legal values of MF are 10, 11, 12, 13, 14, 15, 20, 21, 22, 23, 24, and 25. MF has decimal digits METH and MITER:
 $MF = 10 * METH + MITER$.

METH indicates the basic linear multistep method:
 1 Implicit Adams method.
 2 Method based on backward differentiation formulas (BDF's).

MITER indicates the corrector iteration method:
 0 Functional iteration (no Jacobian matrix is involved).
 1 Chord iteration with a user-supplied full (NEQ by NEQ) Jacobian.
 2 Chord iteration with an internally generated (difference quotient) full Jacobian (using NEQ extra calls to F per df/dy value).
 3 Chord iteration with an internally generated diagonal Jacobian approximation (using one extra call

- to F per df/dy evaluation).
- 4 Chord iteration with a user-supplied banded Jacobian.
 - 5 Chord iteration with an internally generated banded Jacobian (using ML + MU + 1 extra calls to F per df/dy evaluation).

If MITER = 1 or 4, the user must supply a subroutine JAC (the name is arbitrary) as described above under JAC. For other values of MITER, a dummy argument can be used.

Optional Inputs

The following is a list of the optional inputs provided for in the call sequence. (See also Part 2.) For each such input variable, this table lists its name as used in this documentation, its location in the call sequence, its meaning, and the default value. The use of any of these inputs requires IOPT = 1, and in that case all of these inputs are examined. A value of zero for any of these optional inputs will cause the default value to be used. Thus to use a subset of the optional inputs, simply preload locations 5 to 10 in RWORK and IWORK to 0.0 and 0 respectively, and then set those of interest to nonzero values.

Name	Location	Meaning and default value
-----	-----	-----
H0	RWORK(5)	Step size to be attempted on the first step. The default value is determined by the solver.
HMAX	RWORK(6)	Maximum absolute step size allowed. The default value is infinite.
HMIN	RWORK(7)	Minimum absolute step size allowed. The default value is 0. (This lower bound is not enforced on the final step before reaching TCRIT when ITASK = 4 or 5.)
MAXORD	IWORK(5)	Maximum order to be allowed. The default value is 12 if METH = 1, and 5 if METH = 2. (See the MF description above for METH.) If MAXORD exceeds the default value, it will be reduced to the default value. If MAXORD is changed during the problem, it may cause the current order to be reduced.
MXSTEP	IWORK(6)	Maximum number of (internally defined) steps allowed during one call to the solver. The default value is 500.
MXHNIL	IWORK(7)	Maximum number of messages printed (per problem) warning that $T + H = T$ on a step ($H = \text{step size}$). This must be positive to result in a nondefault value. The default value is 10.

Optional Outputs

As optional additional output from SLSODE, the variables listed below are quantities related to the performance of SLSODE which are available to the user. These are communicated by way of the work arrays, but also have internal mnemonic names as shown. Except where stated otherwise, all of these outputs are defined on any successful return from SLSODE, and on any return with ISTATE = -1, -2, -4, -5, or -6. On an illegal input return (ISTATE = -3), they will be unchanged from their existing values (if any), except possibly for TOLSF, LENRW, and LENIW. On any error return,

outputs relevant to the error will be defined, as noted below.

Name	Location	Meaning
-----	-----	-----
HU	RWORK(11)	Step size in t last used (successfully).
HCUR	RWORK(12)	Step size to be attempted on the next step.
TCUR	RWORK(13)	Current value of the independent variable which the solver has actually reached, i.e., the current internal mesh point in t. On output, TCUR will always be at least as far as the argument T, but may be farther (if interpolation was done).
TOLSF	RWORK(14)	Tolerance scale factor, greater than 1.0, computed when a request for too much accuracy was detected (ISTATE = -3 if detected at the start of the problem, ISTATE = -2 otherwise). If ITOL is left unaltered but RTOL and ATOL are uniformly scaled up by a factor of TOLSF for the next call, then the solver is deemed likely to succeed. (The user may also ignore TOLSF and alter the tolerance parameters in any other way appropriate.)
NST	IWORK(11)	Number of steps taken for the problem so far.
NFE	IWORK(12)	Number of F evaluations for the problem so far.
NJE	IWORK(13)	Number of Jacobian evaluations (and of matrix LU decompositions) for the problem so far.
NQU	IWORK(14)	Method order last used (successfully).
NQCUR	IWORK(15)	Order to be attempted on the next step.
IMXER	IWORK(16)	Index of the component of largest magnitude in the weighted local error vector (e(i)/EWT(i)), on an error return with ISTATE = -4 or -5.
LENRW	IWORK(17)	Length of RWORK actually required. This is defined on normal returns and on an illegal input return for insufficient storage.
LENIW	IWORK(18)	Length of IWORK actually required. This is defined on normal returns and on an illegal input return for insufficient storage.

The following two arrays are segments of the RWORK array which may also be of interest to the user as optional outputs. For each array, the table below gives its internal name, its base address in RWORK, and its description.

Name	Base address	Description
-----	-----	-----
YH	21	The Nordsieck history array, of size NYH by (NQCUR + 1), where NYH is the initial value of NEQ. For j = 0,1,...,NQCUR, column j + 1 of YH contains HCUR**j/factorial(j) times the jth derivative of the interpolating polynomial currently representing the solution, evaluated at t = TCUR.
ACOR	LENRW-NEQ+1	Array of size NEQ used for the accumulated corrections on each step, scaled on output to represent the estimated local error in Y on the last step. This is the vector e in the description of the error control. It is defined only on successful return from SLSODE.

Part 2. Other Callable Routines

The following are optional calls which the user may make to gain additional capabilities in conjunction with SLSODE.

Form of call	Function
-----	-----
CALL XSETUN(LUN)	Set the logical unit number, LUN, for output of messages from SLSODE, if the default is not desired. The default value of LUN is 6. This call may be made at any time and will take effect immediately.
CALL XSETF(MFLAG)	Set a flag to control the printing of messages by SLSODE. MFLAG = 0 means do not print. (Danger: this risks losing valuable information.) MFLAG = 1 means print (the default). This call may be made at any time and will take effect immediately.
CALL SSRCOM(RSAV,ISAV,JOB)	Saves and restores the contents of the internal COMMON blocks used by SLSODE (see Part 3 below). RSAV must be a real array of length 218 or more, and ISAV must be an integer array of length 37 or more. JOB = 1 means save COMMON into RSAV/ISAV. JOB = 2 means restore COMMON from same. SSRCOM is useful if one is interrupting a run and restarting later, or alternating between two or more problems solved with SLSODE.
CALL SINTDY(,,,,,) (see below)	Provide derivatives of y, of various orders, at a specified point t, if desired. It may be called only after a successful return from SLSODE. Detailed instructions follow.

Detailed instructions for using SINTDY

The form of the CALL is:

```
CALL SINTDY (T, K, RWORK(21), NYH, DKY, IFLAG)
```

The input parameters are:

T	Value of independent variable where answers are desired (normally the same as the T last returned by SLSODE). For valid results, T must lie between TCUR - HU and TCUR. (See "Optional Outputs" above for TCUR and HU.)
K	Integer order of the derivative desired. K must satisfy $0 \leq K \leq \text{NQCUR}$, where NQCUR is the current order (see "Optional Outputs"). The capability corresponding to $K = 0$, i.e., computing $y(t)$, is already provided by SLSODE directly. Since $\text{NQCUR} \geq 1$, the first derivative dy/dt is always available with SINTDY.
RWORK(21)	The base address of the history array YH.
NYH	Column length of YH, equal to the initial value of NEQ.

The output parameters are:

DKY Real array of length NEQ containing the computed value
 of the Kth derivative of $y(t)$.
IFLAG Integer flag, returned as 0 if K and T were legal,
 -1 if K was illegal, and -2 if T was illegal.
 On an error return, a message is also written.

Part 3. Common Blocks

If SLSODE is to be used in an overlay situation, the user must declare, in the primary overlay, the variables in:

- (1) the call sequence to SLSODE,
- (2) the internal COMMON block /SLS001/, of length 255
(218 single precision words followed by 37 integer words).

If SLSODE is used on a system in which the contents of internal COMMON blocks are not preserved between calls, the user should declare the above COMMON block in his main program to insure that its contents are preserved.

If the solution of a given problem by SLSODE is to be interrupted and then later continued, as when restarting an interrupted run or alternating between two or more problems, the user should save, following the return from the last SLSODE call prior to the interruption, the contents of the call sequence variables and the internal COMMON block, and later restore these values before the next SLSODE call for that problem. In addition, if XSETUN and/or XSETF was called for non-default handling of error messages, then these calls must be repeated. To save and restore the COMMON block, use subroutine SSRCOM (see Part 2 above).

Part 4. Optionally Replaceable Solver Routines

Below are descriptions of two routines in the SLSODE package which relate to the measurement of errors. Either routine can be replaced by a user-supplied version, if desired. However, since such a replacement may have a major impact on performance, it should be done only when absolutely necessary, and only with great caution. (Note: The means by which the package version of a routine is superseded by the user's version may be system-dependent.)

SEWSET

The following subroutine is called just before each internal integration step, and sets the array of error weights, EWT, as described under ITOL/RTOL/ATOL above:

```
SUBROUTINE SEWSET (NEQ, ITOL, RTOL, ATOL, YCUR, EWT)
```

where NEQ, ITOL, RTOL, and ATOL are as in the SLSODE call sequence, YCUR contains the current dependent variable vector, and EWT is the array of weights set by SEWSET.

If the user supplies this subroutine, it must return in EWT(i) (i = 1,...,NEQ) a positive quantity suitable for comparing errors in Y(i) to. The EWT array returned by SEWSET is passed to the SVNORM routine (see below), and also used by SLSODE in the computation of the optional output IMXER, the diagonal Jacobian approximation, and the increments for difference quotient Jacobians.

In the user-supplied version of SEWSET, it may be desirable to use the current values of derivatives of y. Derivatives up to order NQ are available from the history array YH, described above under optional outputs. In SEWSET, YH is identical to the YCUR array, extended to NQ + 1 columns with a column length of NYH and scale factors of $H^{*j}/\text{factorial}(j)$. On the first call for the problem, given by NST = 0, NQ is 1 and H is temporarily set to 1.0. The quantities NQ, NYH, H, and NST can be obtained by including in SEWSET the statements:

```

      REAL RLS
      COMMON /SLS001/ RLS(218),ILS(37)
      NQ = ILS(33)
      NYH = ILS(12)
      NST = ILS(34)
      H = RLS(212)

```

Thus, for example, the current value of dy/dt can be obtained as YCUR(NYH+i)/H (i=1,...,NEQ) (and the division by H is unnecessary when NST = 0).

SVNORM

SVNORM is a real function routine which computes the weighted root-mean-square norm of a vector v:

```

      d = SVNORM (n, v, w)

```

where:

```

n = the length of the vector,
v = real array of length n containing the vector,
w = real array of length n containing weights,
d = SQRT( (1/n) * sum(v(i)*w(i))**2 ).

```

SVNORM is called with n = NEQ and with w(i) = 1.0/EWT(i), where EWT is as set by subroutine SEWSET.

If the user supplies this function, it should return a nonnegative value of SVNORM suitable for use in the error control in SLSODE. None of the arguments should be altered by SVNORM. For example, a user-supplied SVNORM routine might:

- Substitute a max-norm of (v(i)*w(i)) for the rms-norm, or
- Ignore some components of v in the norm, with the effect of suppressing the error control on those components of Y.

```

-----
***REFERENCES  Alan C. Hindmarsh, "ODEPACK, a systematized collection
                of ODE solvers", in Scientific Computing, R. S.
                Stepleman, et al. (Eds.), (North-Holland, Amsterdam,
                1983), pp. 55-64.
***ROUTINES CALLED  SEWSET, SINTDY, RUMACH, SSTODE, SVNORM, XERRWV
***COMMON BLOCKS    SLS001
***REVISION HISTORY  (YYMMDD)

```

791129 DATE WRITTEN
***END PROLOGUE SLSODE

SMAXAF

```
      REAL FUNCTION SMAXAF (ARRAY, IFIRST, ILAST, ISTRID, IMAX)
***BEGIN PROLOGUE  SMAXAF
***PURPOSE  Maximum value in a one-dimensional array.
***LIBRARY   PMATH
***CATEGORY  N5A
***TYPE      SINGLE PRECISION (SMAXAF-S, DMAXAF-D, AMAXF8-8, IMAXAF-I)
***KEYWORDS  MAXIMUM
***AUTHOR   Painter, Jeffrey F., (LLNL)
            Revised by:
            Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine AMAXAF.)
*Usage:
      INTEGER  IFIRST, ILAST, ISTRID, IMAX
      REAL     ARRAY(n), AMAX, SMAXAF
      AMAX = SMAXAF (ARRAY, IFIRST, ILAST, ISTRID, IMAX)

*Arguments:
      ARRAY:IN   Real array to be searched.
                  n, the dimension of the array, must be no less than
                  ILAST.

      IFIRST:IN  First subscript in the array to be searched.

      ILAST :IN  Last subscript in the array to be searched.

      ISTRID:IN  Increment (stride) between successive locations that
                  are to be searched.

      IMAX :OUT  Index of the maximum value in the array, i.e., the
                  ordinal position of the value in the array.

*Function Return Values:
      AMAX :      Maximum value in the array.

*Description:
      SMAXAF finds the maximum value in a one-dimensional real array,
      and returns its index.  In case of multiple maxima, the last
      index found is returned.

*Cautions:
      The array is assumed to be subscripted from 1.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830401  DATE WRITTEN  (J. F. Painter)
      931005  Augmented list of equivalent routines.  (FNF)
      940421  Corrected category.  (FNF)
      940727  Added preprocessor directives for REAL*8 entries.  (FNF)
      951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  SMAXAF
```

SMEANF

```
      REAL FUNCTION SMEANF (A, N)
***BEGIN PROLOGUE  SMEANF
***PURPOSE  Mean of a one-dimensional real array.
***LIBRARY   PMATH
***CATEGORY  L1A
***TYPE      SINGLE PRECISION (SMEANF-S, DMEANF-D, AMEAN8-8)
***KEYWORDS  ELEMENTARY STATISTICS, MEAN
***AUTHOR    Unknown, Name (LLNL/USD/NMG)
              Durst, Mark J. (LLNL/CMRD/SPG)
              Currently responsible:
              Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine AMEANF.)
*Usage:
      INTEGER  N
      REAL    ANS, A(N)
      ANS = SMEANF (A, N)

*Arguments:
      A :IN   Array of input values.
      N :IN   Number of elements in A.

*Function Return Values:
      ANS     The mean of the values in A.

*Description:
      SMEANF calculates the mean of the N values contained in A.

*See Also:
      For a vector of means, see SMEANV.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
              (The above is the date of LCSD-442,Rev.1 and is undoubtedly
              a significant underestimate of the age of this routine.)
      890223  Added SLATEC/LDOC prologue. (FNF)
      890518  Modified sequence numbers to fit in columns 73-80. (FNF)
      920319  Updated with prologue edited 891025 by G. Shaw for manual.
      930930  Converted old UNICOS names to S- or I-names. (DBP)
      931004  Corrected name conversion errors. (FNF)
      931005  Corrected list of equivalent routines and made sure that all
              variables are declared. (FNF)
      931026  Minor change to reduce single/double differences. (FNF)
      940727  Added preprocessor directives for REAL*8 entries. (FNF)
      951010  Corrected LIBRARY line. (FNF)
***END PROLOGUE  SMEANF
```

SMEANV

```
      SUBROUTINE SMEANV (A, N, M, AV)
***BEGIN PROLOGUE  SMEANV
***PURPOSE  Mean vector of a two-dimensional real array.
             Calculates the means of N observations on each of M
             variables.
***LIBRARY    PMATH
***CATEGORY   L1B
***TYPE       SINGLE PRECISION (SMEANV-S, DMEANV-D, MEANV8-8)
***KEYWORDS   ELEMENTARY STATISTICS, MEAN, VECTOR
***AUTHOR    Unknown, Name (LLNL/USD/NMG)
             Durst, Mark J. (LLNL/CMRD/SPG)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
             (Portable version of Cray MATHLIB routine AMEANV.)
*Usage:
      INTEGER  N, M
      REAL    A(N,M), AV(M)
      CALL SMEANV (A, N, M, AV)
```

*Arguments:

```
  A :IN      N by M array of N observations on M variables.
  N :IN      Row dimension of A.
  M :IN      Column dimension of A.
  AV:OUT     Array containing the values of the means, i.e.,
```

$$AV(j) = \sum_{i=1}^N A(i,j) / N, \quad j = 1, \dots, M.$$

*Description:

SMEANV calculates the means of the N observations on each of M variables contained in the columns of A.

The result AV is mathematically equivalent to applying SMEANF to each of the columns of A, but SMEANV should be faster.

```
***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
             (The above is the date of LCSD-442,Rev.1 and is undoubtedly
             a significant underestimate of the age of this routine.)
      951010  Corrected LIBRARY line. (FNF)
***END PROLOGUE  SMEANV
```


SMEDF

```
      REAL FUNCTION SMEDF (A, N, WK)
***BEGIN PROLOGUE  SMEDF
***PURPOSE  Median of a one-dimensional real array.
***LIBRARY   PMATH
***CATEGORY  L1A
***TYPE      SINGLE PRECISION (SMEDF-S, DMEDF-D, AMED8-8)
***KEYWORDS  ELEMENTARY STATISTICS, MEDIAN
***AUTHOR   Unknown, Name (LLNL/USD/NMG)
             Durst, Mark J. (LLNL/CMRD/SPG)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine AMEDF.)
*Usage:
      INTEGER  N
      REAL  ANS, A(N), WK(N)
      ANS = SMEDF (A, N, WK)

*Arguments:
      A  :IN      Array of input values.
      N  :IN      Number of elements in A.
      WK:WORK     Work array of size N.

*Function Return Values:
      ANS : the median of the values in A.

*Description:
      SMEDF calculates the median of the N values contained in A. If N
      is odd, the median is the (N + 1)/2 ordered value. For N even,
      the value is the average of the N/2 and N/2 + 1 ordered values.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
              (The above is the date of LCSD-442,Rev.1 and is undoubtedly
              a significant underestimate of the age of this routine.)
      890223  Added SLATEC/LDOC prologue. (FNF)
      890518  Modified sequence numbers to fit in columns 73-80. (FNF)
      920319  Updated with prologue edited 891025 by G. Shaw for manual.
      930930  Converted old UNICOS names to S- or I-names. (DBP)
      931004  Corrected name conversion errors. (FNF)
      931005  Corrected list of equivalent routines, made sure that all
              variables are declared, and improved comments. (FNF)
      931026  Minor change to reduce single/double differences. (FNF)
      931116  Eliminated two-branch IF statements. (FNF)
      940727  Added preprocessor directives for REAL*8 entries. (FNF)
      951010  Corrected LIBRARY line. (FNF)
***END PROLOGUE  SMEDF
```

SMINAF

```
      REAL FUNCTION SMINAF (ARRAY, IFIRST, ILAST, ISTRID, IMIN)
***BEGIN PROLOGUE  SMINAF
***PURPOSE  Minimum value in a one-dimensional array.
***LIBRARY   PMATH
***CATEGORY  N5A
***TYPE      SINGLE PRECISION (SMINAF-S, DMINAF-D, AMINF8-8, IMINAF-I)
***KEYWORDS  MINIMUM
***AUTHOR   Painter, Jeffrey F., (LLNL)
            Revised by:
            Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine AMINAF.)
*Usage:
      INTEGER  IFIRST, ILAST, ISTRID, IMIN
      REAL     ARRAY(n), AMIN, SMINAF
      AMIN = SMINAF (ARRAY, IFIRST, ILAST, ISTRID, IMIN)

*Arguments:
      ARRAY:IN   Real array to be searched.
                  n, the dimension of the array, must be no less than
                  ILAST.

      IFIRST:IN  First subscript in the array to be searched.

      ILAST :IN  Last subscript in the array to be searched.

      ISTRID:IN  Increment (stride) between successive locations that
                  are to be searched.

      IMIN :OUT  Index of the minimum value in the array, i.e., the
                  ordinal position of the value in the array.

*Function Return Values:
      AMIN :     Minimum value in the array.

*Description:
      SMINAF finds the minimum value in a one-dimensional real array,
      and returns its index.  In case of multiple minima, the last
      index found is returned.

*Cautions:
      The array is assumed to be subscripted from 1.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830401  DATE WRITTEN  (J. F. Painter)
      931005  Augmented list of equivalent routines.  (FNF)
      940421  Corrected category.  (FNF)
      940727  Added preprocessor directives for REAL*8 entries.  (FNF)
      951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  SMINAF
```

SMINMX

```
      SUBROUTINE SMINMX (ARRAY, IFIRST, ILAST, ISTRID, AMIN, AMAX,  
+                      IMIN, IMAX)  
***BEGIN PROLOGUE  SMINMX  
***PURPOSE  Minimum and maximum values in a one-dimensional array.  
***LIBRARY   PMATH  
***CATEGORY  N5A  
***TYPE      SINGLE PRECISION (SMINMX-S, DMINMX-D, AMNMX8-8, IMINMX-I)  
***KEYWORDS  MINIMUM, MAXIMUM  
***AUTHOR   Painter, Jeffrey F., (LLNL)  
            Revised by:  
            Fritsch, Fred N., (LLNL/LC/MSS)  
***DESCRIPTION  
      (Portable version of Cray MATHLIB routine AMINMX.)  
*Usage:  
      INTEGER  IFIRST, ILAST, ISTRID, IMIN, IMAX  
      REAL    ARRAY(n), AMIN, AMAX  
      CALL SMINMX (ARRAY, IFIRST, ILAST, ISTRID, AMIN, AMAX,  
*                IMIN, IMAX)  
  
*Arguments:  
      ARRAY:IN   Real array to be searched.  
                 n, the dimension of the array, must be no less than  
                 ILAST.  
  
      IFIRST:IN  First subscript in the array to be searched.  
  
      ILAST :IN  Last subscript in the array to be searched.  
  
      ISTRID:IN  Increment (stride) between successive locations that  
                 are to be searched (>= 1).  
  
      AMIN :OUT  Minimum value in the array.  
  
      AMAX :OUT  Maximum value in the array.  
  
      IMIN :OUT  Index of the minimum value in the array, i.e., the  
                 ordinal position of the value in the array.  
  
      IMAX :OUT  Index of the maximum value in the array, i.e., the  
                 ordinal position of the value in the array.  
  
*Description:  
      SMINMX finds the minimum and maximum values in a one-dimensional  
      real array, and returns their indices.  In case of multiple  
      extrema, the last index found is returned.  
  
      ISTRID should be greater than or equal to 1.  If ISTRID is less  
      than 1, it is assumed to be 1.  
  
*Cautions:  
      The array is assumed to be subscripted from 1.  
  
***REFERENCES  (NONE)  
***ROUTINES CALLED  (NONE)  
***REVISION HISTORY  (YYMMDD)  
***END PROLOGUE  SMINMX
```

SRANF

```
REAL FUNCTION SRANF()  
***BEGIN PROLOGUE  SRANF  
***PURPOSE  Uniform random-number generator.  
             The pseudorandom numbers generated by SRANF/DRANF/RANF8  
             are uniformly distributed in the open interval (0,1).  
***LIBRARY  PMATH  
***CATEGORY  L6A21  
***TYPE      SINGLE PRECISION (SRANF-S, DRANF-D, RANF8-8)  
***KEYWORDS  RANDOM NUMBER GENERATION, UNIFORM DISTRIBUTION  
***AUTHOR  Fritsch, Fred N., (LLNL/LC/MSS)  
            Original CAL version:  
            Margolies, David, (LLNL/USD/MSS)  
            Durst, Mark J. (LLNL/CMRD/SPG)  
***DESCRIPTION  
            (Portable version of Cray MATHLIB routine RANF.)  
*Usage:  
      REAL  R, SRANF  
      R = SRANF()  
  
*Function Return Values:  
      R          Random number between 0 and 1.  
  
*Description:  
      SRANF generates pseudorandom numbers lying strictly between 0  
      and 1. Each call to SRANF produces a different value, until the  
      sequence cycles after 2**46 calls.  
  
      SRANF is a linear congruential pseudorandom-number generator.  
      The default starting seed is  
          SEED = 4510112377116321(oct) = 948253fc9cd1(hex).  
      The multiplier is 1207264271730565(oct) = 2875a2e7b175(hex).  
  
*See Also:  
      For exponentially distributed random numbers, use SRLGF instead of  
      SRANF.  
      The starting seed for SRANF may be set via RNSSET.  
      The current SRANF seed may be obtained from RNSGET.  
      The SRANF multiplier may be set via RNMSET (changing the  
      multiplier is not recommended).  
      The number of calls to SRANF may be obtained from RNFCNT.  
  
***ROUTINES CALLED  RANF8  
***REVISION HISTORY  (YYMMDD)  
      800325  DATE WRITTEN  
      951009  Minor cosmetic changes.  (FNF)  
      951010  Corrected LIBRARY line.  (FNF)  
***END PROLOGUE  SRANF
```

SRANFV

```
SUBROUTINE SRANFV (N, RANOUT)
***BEGIN PROLOGUE  SRANFV
***PURPOSE  Vector uniform random-number generator.
             Returns a vector of numbers from the SRANF/DRANF/RANF8
             sequence.
***LIBRARY    PMATH
***CATEGORY   L6A21
***TYPE       SINGLE PRECISION (SRANFV-S, DRANFV-D, RANFV8-8)
***KEYWORDS   RANDOM NUMBER GENERATION, UNIFORM DISTRIBUTION, VECTOR
***AUTHOR    Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
             (Portable version of Cray MATHLIB routine RANFV.)
*Usage:
      INTEGER  N
      REAL    RANOUT(n)
      CALL SRANFV (N, RANOUT)

*Arguments:
      N      :IN   Number of random numbers to be generated.
      RANOUT:OUT  Vector of N random numbers between 0 and 1.
                   The actual dimension of RANOUT must satisfy n>=N.

*Description:
      SRANFV generates pseudorandom numbers lying strictly between 0
      and 1. The above call is equivalent to the loop
          DO 10 I=1,N
              RANOUT(I) = SRANF()
          10 CONTINUE
      except that SRANFV may be significantly faster for suitable N.
      (The actual timing is likely to be platform-dependent.)

*See Also:
      Refer to SRANF description for information on restarting the
      sequence and related matters.

***ROUTINES CALLED  SRANF
***REVISION HISTORY  (YYMMDD)
      931011  DATE WRITTEN
      931011  Created portable version that merely calls SRANF.  (FNF)
      931025  Added equivalent routines list.  (FNF)
      940421  Improved purpose.  (FNF)
      940727  Added preprocessor directives for REAL*8 entries.  (FNF)
      951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  SRANFV
```

SRANKS

```
SUBROUTINE SRANKS (A, N, AO, RA, IO, B, ISTAK)
***BEGIN PROLOGUE  SRANKS
***PURPOSE  Ranks of a one-dimensional real array.
***LIBRARY   PMATH
***CATEGORY  L1A
***TYPE      SINGLE PRECISION (SRANKS-S, DRANKS-D, RANKS8-8)
***KEYWORDS  ELEMENTARY STATISTICS, RANKS
***AUTHOR   Unknown, Name, (LLNL/USD/NMG)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
    (Portable version of Cray MATHLIB routine RANKS.)
*Usage:
    INTEGER  N, IO(N), ISTAK(N)
    REAL    A(N), AO(N), RA(N), B(N)
    CALL SRANKS (A, N, AO, RA, IO, B, ISTAK)

*Arguments:
    A      :IN      Array of input values.
    N      :IN      Number of elements in A.
    AO     :OUT     Array containing the values of A ordered.
    RA     :OUT     Array of order N, containing the ranks.
    IO     :WORK    Work array of order N.
    B      :WORK    Work array of order N.
    ISTAK  :WORK    Work array of order N.

*Description:
    SRANKS orders the N values contained in A and calculates their
    ranks.  For ties, the average of the ranks is assigned.

*Accuracy:

*Cautions:
    This routine was formerly known as ORDERS.  Its name was changed
    in March 1991 to avoid conflict with a SCILIB (OMNILIB) routine.

*Portability:

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
    830812  DATE WRITTEN
    931005  Augmented list of equivalent routines, made sure that all
           variables are declared, and improved comments.  (FNF)
    931116  Eliminated two-branch IF statements.  (FNF)
    940727  Added preprocessor directives for REAL*8 entries.  (FNF)
    951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  SRANKS
```

SREFIT

```
SUBROUTINE SREFIT (YDATA, NDATA, MTERMS, WEIGHT, COEFF, RSD2,  
+                WORK, JOB, IERR)  
***BEGIN PROLOGUE  SREFIT  
***PURPOSE  Repeated polynomial fitting.  
             SREFIT(DREFIT) is called after a call of SFITPO(DFITPO) to  
             fit a polynomial of the same or lower degree to the same  
             data or to data in which y has been changed but x left the  
             same.  
***LIBRARY    PMATH  
***CATEGORY   K1A1A2, L8B1B1  
***TYPE       SINGLE PRECISION (SREFIT-S, DREFIT-D, REFIT8-8)  
***KEYWORDS   POLYNOMIAL FITTING, LEAST SQUARES  
***AUTHOR     Painter, Jeffrey F., (LLNL/CMRD)  
             Currently responsible:  
             Fritsch, Fred N., (LLNL/LC/MSS)  
***DESCRIPTION  
             (Portable version of Cray MATHLIB routine REFITP.)  
*Usage:  
    INTEGER  NDATA, MTERMS, JOB, IERR  
    PARAMETER (NWORK = (NDATA+1)*(MTERMS+1) )  
    REAL     YDATA(NDATA), WEIGHT(NDATA), COEFF(MTERMS), RSD2,  
    *        WORK(NWORK)  
  
    CALL SREFIT (YDATA, NDATA, MTERMS, WEIGHT, COEFF, RSD2,  
    *           WORK, JOB, IERR)  
  
*Arguments:  
    YDATA :IN   Array of values (new or old) of the dependent  
                variable, y, of dimension NDATA.  
  
    NDATA :IN   Number of data points.  It must be the same as the  
                NDATA used for SFITPO.  
  
    MTERMS:IN   Number of terms in the polynomial to be found.  It  
                cannot be greater than NTERMS, the number of terms  
                in the polynomial that SFITPO found.  
                If MTERMS > NDATA, the result will be the coefficients  
                of an interpolating polynomial of degree NDATA-1, and  
                COEFF(j) = 0 for j > NDATA.  
  
    WEIGHT:IN   Optional weight array.  It must be the same as in  
                the SFITPO call.  
  
    COEFF :OUT   Array containing the MTERMS coefficients of the  
                polynomial.  
  
    RSD2  :OUT   Sum of the squares of the (weighted) residuals  
                corresponding to COEFF.  
  
    WORK :WORK   Must be exactly the same array as in the previous  
                call of SFITPO or SREFIT; no changes may be made by  
                the calling program.  As in SFITPO, WORK contains the  
                residuals R(i) in its first NDATA entries if JOB is  
                nonzero.  
  
    JOB  :IN     Residuals-computation flag:
```

non-0 Residuals are computed and output in WORK.
0 Residuals are not completely computed,
 although RSD2 is computed. (This option will
 more efficient if the R(i) are not required.)

IERR :OUT Error flag. On normal termination, IERR = 0.

Fatal errors:

- (1) SQRSL returned INFO=IERR: 0 < IERR <= NTERMS
A singular matrix has been detected (same meaning
as in SFITPO). SREFIT should not be called if
SFITPO returned IERR > 0.
- (2) MTERMS > NTERMS: IERR = -1
COEFF has not been computed in either case.

*Description:

SREFIT is called, after a call of SFITPO, to fit a polynomial of the same or lower degree to the same data or to data in which y has been changed but x left the same as in the SFITPO call.

SREFIT provides the same output as would a second call of SFITPO, but SREFIT is more efficient. SREFIT may be called any number of times, as long as the contents of WORK are not disturbed.

*Portability:

This routine calls the LINPACK routine SQRSL, and BLAS (Basic Linear Algebra Subprograms) SCOPY, SDOT.

*See Also:

See SFITPO for additional information.

***SEE ALSO SFITPO
***REFERENCES (NONE)
***ROUTINES CALLED SCOPY, SDOT, SQRSL
***REVISION HISTORY (YYMMDD)
***END PROLOGUE SREFIT

SRLGF

```
      REAL FUNCTION SRLGF()  
***BEGIN PROLOGUE  SRLGF  
***PURPOSE  Exponential random-number generator.  
             The pseudorandom numbers generated by SRLGF/DRLGF/RLGF8  
             are drawn from the exponential distribution with mean 1.  
***LIBRARY  PMATH  
***CATEGORY  L6A5  
***TYPE      SINGLE PRECISION (SRLGF-S, DRLGF-D, RLGF8-8)  
***KEYWORDS  RANDOM NUMBER GENERATION, EXPONENTIAL DISTRIBUTION  
***AUTHOR  Fritsch, Fred N., (LLNL/LC/MSS)  
            Original CAL version:  
            Margolies, David, (LLNL/USD/MSS)  
            Durst, Mark J. (LLNL/CMRD/SPG)  
***DESCRIPTION  
            (Portable version of Cray MATHLIB routine RLGF.)  
*Usage:  
      REAL  R  
      R = SRLGF()  
  
*Function Return Values:  
      R      A random number drawn from the exponential distribution  
            with mean 1.  
  
*Description:  
      SRLGF takes the natural logarithm of uniform random numbers.  
      SRLGF() should be used in place of the expression -LOG(SRANF()).  
      Each call to SRLGF produces a different value, until the sequence  
      cycles after 2**46 calls.  
  
      SRLGF uses a linear congruential pseudorandom-number generator  
      which is identical to SRANF except that the default starting seed  
      is different:  
          SEED = 7315512527213717(oct) = ecda555d17cf(hex).  
      The multiplier is 1207264271730565(oct) = 2875a2e7b175(hex).  
      The SRLGF/DRLGF/RLGF8 sequence is independent of that generated  
      by SRANF/DRANF/RANF8.  
  
*Cautions:  
      Note that if you are using both SRANF and SRLGF, stopping and  
      restarting both sequences will require calling both RNSGET/RNSSET  
      and RLSGET/RLSSET.  
  
***ROUTINES CALLED  RLGF8  
***REVISION HISTORY  (YYMMDD)  
      800325  DATE WRITTEN  
***END PROLOGUE  SRLGF
```

SSRCOM

```
      SUBROUTINE SSRCOM (RSAV, ISAV, JOB)
***BEGIN PROLOGUE  SSRCOM
***PURPOSE  Save/restore ODEPACK COMMON blocks.
***LIBRARY   PMATH (ODEPACK)
***CATEGORY  11C
***TYPE      SINGLE PRECISION (SSRCOM-S, DSRCOM-D, SRCOM8-8)
***AUTHOR   Hindmarsh, Alan C., (LLNL)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine SRCOM.)

This routine saves or restores (depending on JOB) the contents of
the COMMON block SLS001, which is used internally
by one or more ODEPACK solvers.

RSAV = real array of length 218 or more.
ISAV = integer array of length 37 or more.
JOB  = flag indicating to save or restore the COMMON blocks:
      JOB = 1 if COMMON is to be saved (written to RSAV/ISAV)
      JOB = 2 if COMMON is to be restored (read from RSAV/ISAV)
      A call with JOB = 2 presumes a prior call with JOB = 1.

***SEE ALSO  SLSODE
***ROUTINES CALLED  (NONE)
***COMMON BLOCKS  SLS001
***REVISION HISTORY  (YYMMDD)
      791129  DATE WRITTEN
      890501  Modified prologue to SLATEC/LDOC format.  (FNF)
      890503  Minor cosmetic changes.  (FNF)
      921116  Deleted treatment of block /EH0001/.  (ACH)
      930801  Reduced Common block length by 2.  (ACH)
      930809  Renamed to allow single/double precision versions.  (ACH)
      940315  Added REAL*8 name to C***TYPE line.  (FNF)
      940727  Added preprocessor directives for REAL*8 entries.  (FNF)
      941011  Changed to user-callable.  (FNF)
      951010  Corrected LIBRARY line.  (FNF)
***END PROLOGUE  SSRCOM
```

SSTDEV

```
      REAL FUNCTION SSTDEV (A, N, IND)
***BEGIN PROLOGUE  SSTDEV
***PURPOSE  Standard deviation of a one-dimensional real array.
***LIBRARY   PMATH
***CATEGORY  L1A
***TYPE      SINGLE PRECISION (SSTDEV-S, DSTDEV-D, STDEV8-8)
***KEYWORDS  ELEMENTARY STATISTICS, STANDARD DEVIATION
***AUTHOR   Unknown, Name (LLNL/USD/NMG)
             Durst, Mark J. (LLNL/CMRD/SPG)
             Currently responsible:
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
      (Portable version of Cray MATHLIB routine STDEVF.)
*Usage:
      INTEGER  N, IND
      REAL    ANS, A(N)
      ANS = SSTDEV (A, N, IND)

*Arguments:
      A  :IN  Array of input values.
      N  :IN  Number of elements in A.
      IND:IN  Job-control flag:
               0  Divide the adjusted sum of squares by N - 1,
                   producing the usual standard-deviation calculation.
              non-0 Divide by N.

*Function Return Values:
      ANS      The standard deviation of the values in A.

*Description:
      SSTDEV calculates the standard deviation of the N values contained
      in A, as modified by IND.

*See Also:
      For a vector of standard deviations, see SCOVAR.

***REFERENCES  (NONE)
***ROUTINES CALLED  (NONE)
***REVISION HISTORY  (YYMMDD)
      830812  DATE WRITTEN
              (The above is the date of LCSD-442,Rev.1 and is undoubtedly
              a significant underestimate of the age of this routine.)
***END PROLOGUE  SSTDEV
```

SZERO

```
SUBROUTINE SZERO (F, B, C, ABSERR, RELERR, IFLAG)
***BEGIN PROLOGUE  SZERO
***PURPOSE  Find a root x of a nonlinear equation  $F(x) = 0$ .
             A search interval (b,c) must be supplied such that
              $F(b)*F(c) \leq 0$ .
***LIBRARY    PMATH
***CATEGORY   F1B
***TYPE       SINGLE PRECISION (SZERO-S, DZERO-D, ZERO8-8)
***KEYWORDS   ZEROFINDING, NONLINEAR EQUATIONS, SECANT METHOD,
             BISECTION METHOD
***AUTHOR    Leonard, L. J., (LLNL)
             Fritsch, Fred N., (LLNL/LC/MSS)
***DESCRIPTION
             (Portable version of Cray MATHLIB routine ZEROIN.)
*Usage:
    INTEGER  IFLAG
    REAL    F, B, C, ABSERR, RELERR
    EXTERNAL F
    CALL SZERO (F, B, C, ABSERR, RELERR, IFLAG)

*Arguments:
    F :EXT      Name of a function subprogram defining a continuous
                real function of a single real variable x. The
                calling program must declare the function to be
                EXTERNAL.

    B :INOUT    Input:  Lower bound of the search interval (B,C).
                Output: The better approximation to a root, for B
                and C are redefined so that
                 $ABS(F(B)) \leq ABS(F(C))$ .

    C :INOUT    Input:  Upper bound of the search interval (B,C).
                Output: The value of C is not necessarily close to
                B and should be disregarded (see B above).

    ABSERR:IN   Roughly the maximum difference allowed between B
                and C.  If zero is a possible root, do not use
                ABSERR = 0.

    RELERR:IN   Roughly the maximum relative error allowed between
                B and C; i.e., the degree of accuracy required in
                the root.

    IFLAG:INOUT Input:
                 $\geq 6$   The maximum number of function evaluations
                allowed.
                 $< 6$   The maximum number of evaluations is 100.

                Output:
                1   $F(B) * F(C) < 0$ , and the stopping criterion
                    $ABS(B - C) \leq 2.0 * (RELERR * ABS(B) + ABSERR)$ 
                   is met.
                2  B is found such that  $F(B) = 0$ . The interval
                   (B,C) may or may not have satisfied the stopping
                   criterion.
                3   $ABS(F(B))$  exceeds the absolute values of the
```

- function at the original input values of B and C;
i.e., the values found by SZERO are "worse" than
those supplied in the call. In this case, it is
likely that B is near a pole of the function.
- 4 No odd-order zero was found in the interval. A
local minimum may have been obtained.
 - 5 The stopping criterion is not met within the
specified number of function evaluations.

***Description:**

SZERO finds a root x of the nonlinear equation $F(x) = 0$. Normal input consists of a continuous function F and an initial search interval (B,C) that brackets the desired zero of F ; i.e., $F(B) * F(C) \leq 0$.

Each iteration finds new values of B and C such that the interval (B,C) is shrunk, and $F(B) * F(C) \leq 0$. The stopping criterion is

$$\text{ABS}(B - C) \leq 2.0 * (\text{RELERR} * \text{ABS}(B) + \text{ABSERR})$$

SZERO is a slightly modified version of the subroutine SZERO by Shampine and Allen (see Ref. 2). The method used is a combination of bisection and the secant iteration.

***Cautions:**

F is assumed to be a continuous real-valued function. The algorithm in SZERO assumes that F has exactly one zero in the interval $[B,C]$. If, in fact, F has an odd number of zeros, SZERO will zero in on one of them, giving no indication that there may be more.

***ROUTINES CALLED RUMACH

***REVISION HISTORY (YYMMDD)

940425 Added "EXTERNAL F" statement for certain compilers. (FNF)

940727 Added preprocessor directives for REAL*8 entries. (FNF)

951010 Corrected LIBRARY line. (FNF)

***END PROLOGUE SZERO

XERROR

```
SUBROUTINE XERROR (MESSG, NMESSG, NERR, LEVEL)
***BEGIN PROLOGUE  XERROR
***PURPOSE  Process an error (diagnostic) message.
***LIBRARY   PMATH
***TYPE      ALL (XERROR-A)
***CATEGORY  R3C
***KEYWORDS  ERROR
***AUTHOR    JONES, R. E., (SNLA)
***DESCRIPTION
```

Abstract

XERROR processes a diagnostic message, in a manner determined by the value of LEVEL and the current value of the library error control flag, KONTRL.
(See subroutine XSETF for details.)

Description of Parameters

--Input--

MESSG - the Hollerith message to be processed, containing no more than 72 characters.
NMESSG- the actual number of characters in MESSG.
NERR - the error number associated with this message.
NERR must not be zero.
LEVEL - error category.
=2 means this is an unconditionally fatal error.
=1 means this is a recoverable error. (I.e., it is non-fatal if XSETF has been appropriately called.)
=0 means this is a warning message only.
=-1 means this is a warning message which is to be printed at most once, regardless of how many times this call is executed.

Examples

```
CALL XERROR('SMOOTH -- NUM WAS ZERO.',23,1,2)
CALL XERROR('INTEG -- LESS THAN FULL ACCURACY ACHIEVED.',
1          43,2,1)
CALL XERROR('ROOTER -- ACTUAL ZERO OF F FOUND BEFORE INTERVAL F
1ULLY COLLAPSED.',65,3,0)
CALL XERROR('EXP -- UNDERFLOWS BEING SET TO ZERO.',39,1,-1)
```

Written by Ron Jones, with SLATEC Common Math Library Subcommittee

```
***REFERENCES  JONES R.E., KAHANER D.K., 'XERROR, THE SLATEC ERROR-
HANDLING PACKAGE', SAND82-0800, SANDIA LABORATORIES,
1982.
***ROUTINES CALLED  XERRWV
***REVISION HISTORY (YYMMDD)
***END PROLOGUE  XERROR
```

XERRWD

```
      SUBROUTINE XERRWD (MSG, NMES, NERR, LEVEL, NI, I1, I2, NR, R1, R2)
***BEGIN PROLOGUE  XERRWD
***PURPOSE  Write error message with values.
***LIBRARY   PMATH
***CATEGORY  R3C
***TYPE      DOUBLE PRECISION (XERRWV-S, XERRWD-D)
***AUTHOR   Hindmarsh, Alan C., (LLNL)
***DESCRIPTION
```

Subroutines XERRWD, XSETF, XSETUN, and the function routine IXSAV, as given here, constitute a simplified version of the SLATEC error handling package.

All arguments are input arguments.

```
MSG      = The message (character array).
NMES     = The length of MSG (number of characters).
NERR     = The error number (not used).
LEVEL    = The error level..
           0 or 1 means recoverable (control returns to caller).
           2 means fatal (run is aborted--see note below).
NI       = Number of integers (0, 1, or 2) to be printed with message.
I1,I2    = Integers to be printed, depending on NI.
NR       = Number of reals (0, 1, or 2) to be printed with message.
R1,R2    = Reals to be printed, depending on NR.
```

Note.. this routine is machine-dependent and specialized for use in limited context, in the following ways..

1. The argument MSG is assumed to be of type CHARACTER, and the message is printed with a format of (1X,A).
2. The message is assumed to take only one line. Multi-line messages are generated by repeated calls.
3. If LEVEL = 2, control passes to the statement STOP to abort the run. This statement may be machine-dependent.
4. R1 and R2 are assumed to be in double precision and are printed in D21.13 format.

```
***ROUTINES CALLED  IXSAV
***REVISION HISTORY (YYMMDD)
  920831  DATE WRITTEN
  921118  Replaced MFLGSV/LUNSAV by IXSAV. (ACH)
  930329  Modified prologue to SLATEC format. (FNF)
  930407  Changed MSG from CHARACTER*1 array to variable. (FNF)
  930922  Minor cosmetic change. (FNF)
  941011  Changed to user-callable. (FNF)
  951010  Corrected LIBRARY line. (FNF)
***END PROLOGUE  XERRWD
```

XERRWV

```
      SUBROUTINE XERRWV (MSG, NMES, NERR, LEVEL, NI, I1, I2, NR, R1, R2)
***BEGIN PROLOGUE  XERRWV
***PURPOSE  Write error message with values.
***LIBRARY   PMATH
***CATEGORY  R3C
***TYPE      SINGLE PRECISION (XERRWV-S, XERRWD-D)
***AUTHOR   Hindmarsh, Alan C., (LLNL)
***DESCRIPTION
```

Subroutines XERRWV, XSETF, XSETUN, and the function routine IXSAV, as given here, constitute a simplified version of the SLATEC error handling package.

All arguments are input arguments.

```
MSG      = The message (character array).
NMES     = The length of MSG (number of characters).
NERR     = The error number (not used).
LEVEL    = The error level..
           0 or 1 means recoverable (control returns to caller).
           2 means fatal (run is aborted--see note below).
NI       = Number of integers (0, 1, or 2) to be printed with message.
I1,I2    = Integers to be printed, depending on NI.
NR       = Number of reals (0, 1, or 2) to be printed with message.
R1,R2    = Reals to be printed, depending on NR.
```

Note.. this routine is machine-dependent and specialized for use in limited context, in the following ways..

1. The argument MSG is assumed to be of type CHARACTER, and the message is printed with a format of (1X,A).
2. The message is assumed to take only one line. Multi-line messages are generated by repeated calls.
3. If LEVEL = 2, control passes to the statement STOP to abort the run. This statement may be machine-dependent.
4. R1 and R2 are assumed to be in single precision and are printed in E21.13 format.

```
***ROUTINES CALLED  IXSAV
***REVISION HISTORY (YYMMDD)
   791129  DATE WRITTEN
***END PROLOGUE  XERRWV
```


XSETF

```
      SUBROUTINE XSETF (MFLAG)
***BEGIN PROLOGUE  XSETF
***PURPOSE  Reset the error print control flag.
***LIBRARY   PMATH
***CATEGORY  R3A
***TYPE      ALL (XSETF-A)
***KEYWORDS  ERROR CONTROL
***AUTHOR   Hindmarsh, Alan C., (LLNL)
***DESCRIPTION

      XSETF sets the error print control flag to MFLAG:
      MFLAG=1 means print all messages (the default).
      MFLAG=0 means no printing.

***SEE ALSO  XERMSG, XERRWD, XERRWV
***REFERENCES  (NONE)
***ROUTINES CALLED  IXSAV
***REVISION HISTORY  (YMMDD)
      921118  DATE WRITTEN
      930329  Added SLATEC format prologue. (FNF)
      930407  Corrected SEE ALSO section. (FNF)
      930922  Made user-callable, and other cosmetic changes. (FNF)
      951010  Corrected LIBRARY line. (FNF)
***END PROLOGUE  XSETF
```

XSETUN

```
      SUBROUTINE XSETUN (LUN)
***BEGIN PROLOGUE  XSETUN
***PURPOSE  Reset the logical unit number for error messages.
***LIBRARY   PMATH
***CATEGORY  R3B
***TYPE      ALL (XSETUN-A)
***KEYWORDS  ERROR CONTROL
***DESCRIPTION

      XSETUN sets the logical unit number for error messages to LUN.

***AUTHOR  Hindmarsh, Alan C., (LLNL)
***SEE ALSO  XERMSG, XERRWD, XERRWV
***REFERENCES  (NONE)
***ROUTINES CALLED  IXSAV
***REVISION HISTORY  (YYMMDD)
      921118  DATE WRITTEN
      930329  Added SLATEC format prologue. (FNF)
      930407  Corrected SEE ALSO section. (FNF)
      930922  Made user-callable, and other cosmetic changes. (FNF)
      951010  Corrected LIBRARY line. (FNF)
***END PROLOGUE  XSETUN
```

Disclaimer

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government thereof, and shall not be used for advertising or product endorsement purposes.

(C) Copyright 2002 The Regents of the University of California. All rights reserved.

Keyword Index

To see an alphabetical list of keywords for this document, consult the [next section](#) (page 150).

Keyword	Description
<u>entire</u>	This entire document.
<u>title</u>	The name of this document.
<u>function</u>	Topics covered in this document.
<u>availability</u>	Where these programs run.
<u>who</u>	Who to contact for assistance.
<u>introduction</u>	Role and goals of this document.
<u>background</u>	PMATH's relation to MATHLIB.
<u>availability-2</u>	PMATH, MATHLIB, SLATEC access chart.
<u>design-principles</u>	How PMATH was planned.
<u>names</u>	Naming scheme for PMATH, MATHLIB.
<u>random-numbers-0</u>	PMATH's support for random nums.
<u>other-routines-0</u>	Constants and conversions in PMATH.
<u>mathlib-categories</u>	Routines categorized, MATHLIB names.
<u>mathlib-included</u>	MATHLIB rtns included in PMATH.
	These categories divide the library:
	<u>elementary-functions-1</u>
	<u>random-numbers-1</u>
	<u>max-min-1</u>
	<u>table-look-up-1</u>
	<u>statistics-1</u>
	<u>linear-algebra-1</u>
	<u>root-finders-1</u>
	<u>interpolation-1</u>
	<u>differential-equations-1</u>
	<u>other-routines-1</u>
	<u>error-procedures-1</u>
<u>mathlib-omitted</u>	MATHLIB rtns omitted from PMATH.
	These categories have excluded rtns:
	<u>elementary-functions-2</u>
	<u>random-numbers-2</u>
	<u>max-min-2</u>
	<u>differential-equations-2</u>
	<u>error-procedures-2</u>
<u>name-chart</u>	PMATH-MATHLIB name-conversion chart.
<u>pmath-categories</u>	Routines categorized, PMATH names.
<u>task-list</u>	PMATH rtns grouped by function.
	These categories divide the library:
	<u>elementary-functions</u>
	<u>random-numbers</u>
	<u>max-min</u>
	<u>table-look-up</u>
	<u>statistics</u>
	<u>linear-algebra</u>
	<u>root-finders</u>
	<u>interpolation</u>
	<u>differential-equations</u>

<u>other-routines</u>	
<u>error-procedures</u>	
<u>added-routines</u>	Routines new to PMATH.
<u>vectorized-ranf</u>	Portable version of RANFV.
<u>seed-passing</u>	Moving random-number seeds (2 rtns).
<u>pmath-routines</u>	Alphabetized PMATH routine prologs. (See <u>next section</u> for list.)
<u>index</u>	The structural index of keywords.
<u>a</u>	The alphabetical index of keywords.
<u>date</u>	The latest changes to this document.
<u>revisions</u>	The complete revision history.

Alphabetical List of Keywords

All PMATH routines described in this manual are listed here alphabetically by root name (RIGHT-most column), with a link to the single precision (S-name) prolog if one exists. Corresponding MATHLIB routine names appear in the LEFT-most column, for reference.

MATHLIB (UNICOS) name	S-name	-PMATH- D-name	REAL*8 name
-----	-----	-----	-----
AAAAAA	---	---	<u>AAAAAA</u>
AMAXAF	SMAXAF	DMAXAF	<u>AMAXF8</u>
AMEANF	SMEANF	DMEANF	<u>AMEAN8</u>
AMEDF	SMEDF	DMEDF	<u>AMED8</u>
AMINAF	SMINAF	DMINAF	<u>AMINF8</u>
AMINMX	SMINMX	DMINMX	<u>AMNMX8</u>
CONSTANT	SCONST	DCONST	<u>CONST8</u>
CORRV	SCORRV	DCORRV	<u>CORRV8</u>
COVARV	SCOVAR	DCOVAR	<u>COVAR8</u>
---	---	---	<u>CV16TO64</u>
---	---	---	<u>CV64TO16</u>
FITPOL	SFITPO	DFITPO	<u>FITPO8</u>
MAXAF	---	---	<u>IMAXAF</u>
MINAF	---	---	<u>IMINAF</u>
MINMX	---	---	<u>IMINMX</u>
IUMACH	---	---	<u>IUMACH</u>
LDF	LDFS	LDFD	<u>LDF8</u>
LSODE	SLSODE	DLNODE	<u>LSODE8</u>
LUF	LUFS	LUFD	<u>LUF8</u>
LUG	LUGS	LUGD	<u>LUG8</u>
AMEANV	SMEANV	DMEANV	<u>MEANV8</u>
RANF	SRANF	DRANF	<u>RANF8</u>
RANFV	SRANFV	DRANFV	<u>RANFV8</u>
RANKS	SRANKS	DRANKS	<u>RANKS8</u>
REFITP	SREFIT	DREFIT	<u>REFIT8</u>
RLGCNT	---	---	<u>RLFCNT</u>
RLGF	SRLGF	DRLGF	<u>RLGF8</u>
RLGMSSET	---	---	<u>RLMSET</u>
RLGGET	---	---	<u>RLSGET</u>
RLGSET	---	---	<u>RLSSET</u>
RNCOUNT	---	---	<u>RNFCNT</u>
RNMUSET	---	---	<u>RNMSET</u>
RANGET	---	---	<u>RNSGET</u>
RANSET	---	---	<u>RNSSET</u>
STDEVF	SSTDEV	DSTDEV	<u>STDEV8</u>
RUMACH	RUMACH	DUMACH	<u>UMACH8</u>
XERROR	---	---	<u>XERROR</u>

XERRWV	XERRWV	XERRWD	<u>XERRWV</u>
XSETF	---	---	<u>XSETF</u>
XSETUN	---	---	<u>XSETUN</u>
ZEROIN	SZERO	DZERO	<u>ZERO8</u>

Date and Revisions

Revision Date -----	Keyword Affected -----	Description of Change -----
25Nov02	<u>background</u> <u>availability-2</u>	Pathname clarified. Compaqs, Linux now included.
21Mar00	entire	CRAYs retired; all CRAY references revised or deleted.
24Jul97	entire	First edition of PMATH manual.

TRG (25Nov02)

UCRL-WEB-201525

LLNL Privacy and Legal Notice (URL: <http://www.llnl.gov/disclaimer.html>)

TRG (25Nov02) Contact on the OCF: lc-hotline@llnl.gov, on the SCF: lc-hotline@pop.llnl.gov